MacTech Magazine
Vol. 13, No. 5 • May 1997

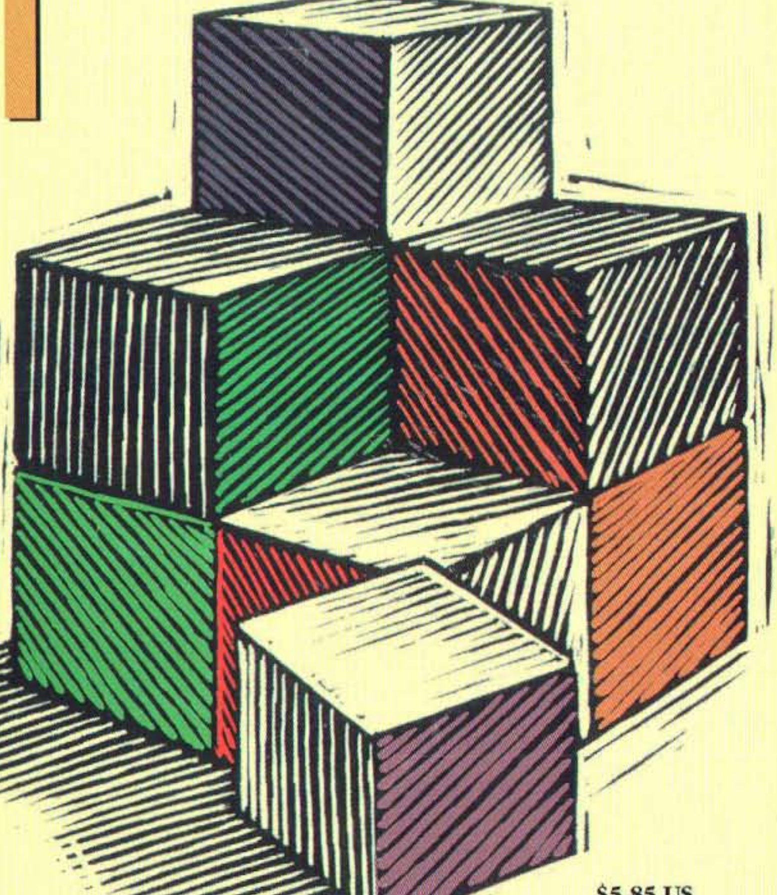
For Macintosh Programmers & Developers

MacTech™

NEW! develop Section by Apple Computer, Inc.

Includes Special develop Section by Apple Computer, Inc.

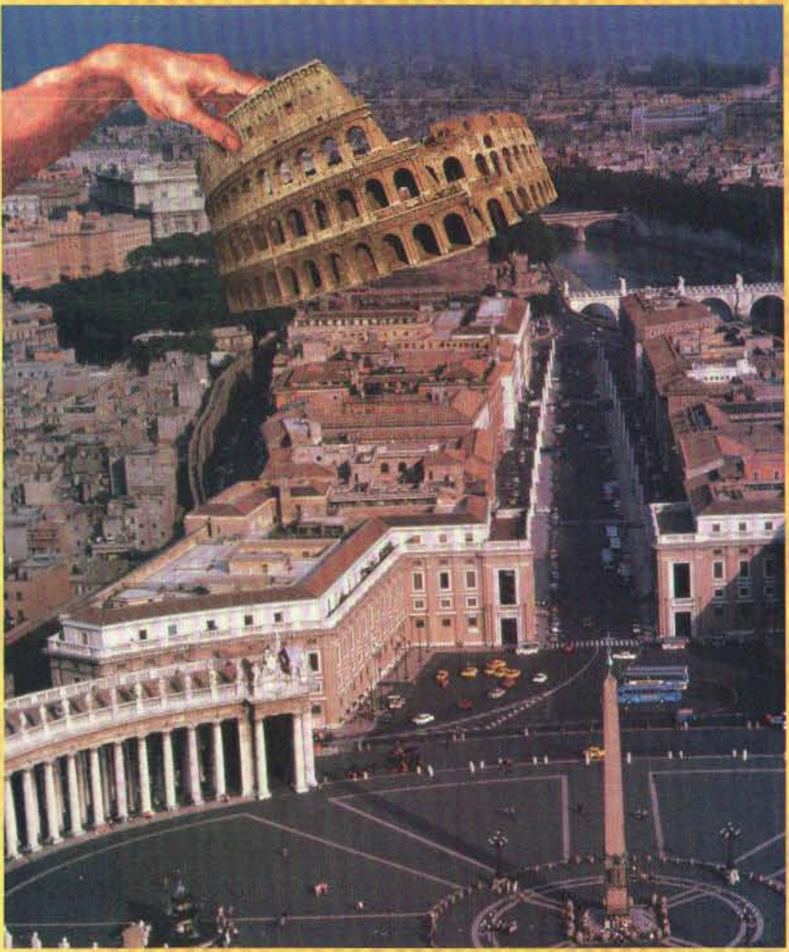
INSIDE OPENSTEP



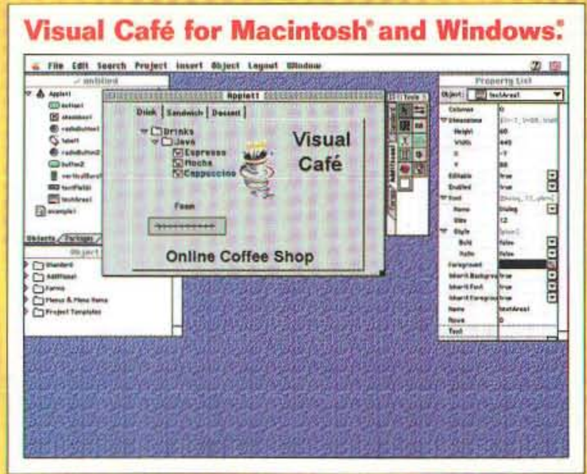
- **Building an OPENSTEP Application**
- **An Introduction to WebObjects**
- **The Metrowerks Road to Rhapsody**

$5.85 US
$6.95 Canada
ISSN 1067-8360
Printed in U.S.A.

Build Rome in a day. It's easy with Symantec Visual Café™ for Java.™



Visual Café for Macintosh® and Windows.®



Visual Java Development

Develop faster. Compile faster. Debug faster. Realize your Java™ dreams faster.

Symantec, the creator of the first full-featured Java development environment, now unleashes the first Rapid Application Development environment for Java developers:

Symantec Visual Café.™

Visual Café comes complete with an extensible component library with all of the building blocks you need for your application.

Simply drag and drop a component onto a form. Our Interaction Wizard lets you visually specify all the actions and events. And then Visual Café automatically generates the Java code for you.

Thanks to our exclusive two-way programming you can add or modify the code at the source level, too.

So you'll be whipping out those application prototypes at speeds you can only dream of now.

Imagine building all of your forms visually. Or building your entire user interface without writing one single line of code!

As an added bonus, Symantec's Just In Time (JIT) compiler (included in Netscape's new Navigator™) runs your Java applications faster than any other browser or Java virtual machine on the planet.

So get your hands on the hottest new development tool for Java right away.

For more information, call us at 1-800-453-1059 ext. 9H29 or visit our Web site at cafe.symantec.com. Or pick up a copy at Fry's, MicroCenter, CompUSA or Egghead.

JAVA COMPATIBLE

Mac OS

Designed for Microsoft® Windows®95

SYMANTEC.®

Eddy Award Winner** for **Best New Developer Tool
– MacUser Editors Choice Awards, 1993

"A distinct improvement over ResEdit."
– MacTech / MacTutor

"Resorcerer's data template system is amazing!"
– Bill Goodman, author of Compact Pro

"Nuke ResEdit! Resorcerer is mission-critical for us."
– Dave Winer, Userland Frontier

"The color pixel editors are wonderful! A work of art!"
– Dave Winzler, author of Microseeds Redux

"Every Macintosh developer should own a copy of Resorcerer."
– Leonard Rosenthol, Aladdin Systems

"Resorcerer will pay for itself many times over in saved time and effort."
– MacUser review

"The template that disassembles 'PICT's is awesome!"
– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensible in its own creation!"
– Doug McKenna, author of Resorcerer

"...a wealth of time-saving tools."
MacUser Review, Dec. 1992

MACUSER EDITORS' CHOICE AWARD 1993 WINNER

RESORCERER®

Version 1.2.4

The Resource Editor for the Macintosh Wizard

ORDERING INFO

Needs: ≥Mac Plus, ≥ Sys 4.2, 1MB
Likes: ≥Mac Plus, ≥ Sys 7.0, 2MB
32-bit clean, AU/X compatible

Price: $256 (decimal)
(Educational, quantity, or other discounts available)

Includes: 500 page manual
60-day Money-Back Guarantee
Domestic UPS ground shipping

Payment: Check, PO's, or Visa/MC

Extras (call us):
COD, FedEx, UPS Blue/Red,
International Shipping

Downloadable Demos/Updaters:
AppleLink: Software Sampler
AOL: Software Libs/Development
CompuServe: MACDEV/Tools
or call us.

New 1.2 Features:

- **New 'cicn', 'ppat', 'crsr', 'acur', 'pltt', 'clut' editors**
- **Powerful icon family editing (all 9 icon types)**
- **Color pixel anti-aliasing, dithering, and lots more**
- **Complete 'PICT' disassembly and reassembly**
- **Resource sorting; ROM resource browsing**
- **120 template field parsing types now supported**
- **New insertion & deletion template field types**
- **Text-only 'PICT' resources**
- **Lots of improvements throughout**

- Easier, faster, more Mac-like, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Help, AppleEvent, OCE, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Well-designed, helpful developer tools being added all the time
- Relied on by thousands of Macintosh developers around the world

MATHEMÆSTHETICS, INC.
P.O. Box 298 • Boulder • CO • 80306-0298 • USA
Phone: (303) 440-0707 • Fax: (303) 440-0504
AppleLink/AmericaOnline: RESORCERER • Internet: resorcerer@aol.com

For Macintosh Programmers & Developers

MacTech™ MAGAZINE

A publication of XPLAIN CORPORATION

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 800-MACDEV-1

DEPARTMENTS	E-Mail/URL
Orders, Circulation, & Customer Service	cust_service@devdepot.com
Press Releases	press_releases@mactech.com
Ad Sales	ad_sales@mactech.com
Editorial	editorial@mactech.com
Programmer's Challenge	prog_challenge@mactech.com
Online Support	online@mactech.com
Accounting	accounting@mactech.com
Marketing	marketing@mactech.com
General	info@mactech.com
Web Site (articles, info, URLs and more...)	http://www.mactech.com

MACTECH MAGAZINE

MacTech Magazine is grateful to the following individuals who contribute on a regular basis. We encourage others to share the technology.

We are dedicated to the distribution of useful programming information without regard to Apple's developer status.

*For information on submitting articles, ask us for our **writer's kit** which includes the terms and conditions upon which we publish articles.*

Editorial Board of Advisors

Scott T Boyd, Jordan J. Mattson, Jim Straus, and Jon Wiederspan

Editors

Publisher • Neil Ticktin
Editor-in-Chief • Eric Gundrum
Managing Editor • Jessica Courtney
Editorial Intern • Michelle Shin
Online Support • Nick "nick.c" DeMello

Contributing and Technical Editors

Java • Will Iverson, Apple Computer, Inc.
Internet • Carl de Cordova, Apple Computer, Inc.
MacOS 8 • Steve Kiene, Mindvision
MacDev-1™ • Rich Siegel, Bare Bones Software, Inc.
MagicCap/TeleScript • Richard Clark
Components • Tantek Çelik, 6prime Corporation
Performance Programming • Jim Gochee, Connectix
Product Reviews • Ed Ringel

Regular Columnists

Getting Started • Dave Mark
Programmer's Challenge • Bob Boonstra
Symantec Top 10 • Symantec Technical Support
From the Factory Floor • Dave Mark, Metrowerks
Tips & Tidbits • Steve Sisak
MacTech Online/URLs • Nick "nick.c" DeMello

XPLAIN CORPORATION

Chief Executive Officer • Neil Ticktin
Chief Operating Officer • Andrea J. Sniderman
Advertising Executive • Ruth Subrin
Marketing Manager • Susan M. Worley
Customer Relations • Erik Davidson, Stephanie Long, Susan Pomrantz
Accounting • Dennise Gant, Jan Webber
Network Administrator • Donal Corcoran

Financial Services • Shin Financial Services
Art Direction/Production • InfoGraphix

Board of Advisors • Steven Geller, Blake Park, and Alan Carsrud

PRINTED WITH SOYINK™

This publication is printed on paper with recycled content.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

TABLE OF CONTENTS

For Macintosh Programmers & Developers

MacTech™ MAGAZINE

May 1997 • Volume 13, No. 5

VIEWPOINT

By Eric Gundrum

Have You Put Your Development on Hold?

Do you have your development projects on hold while you wait for more information about Rhapsody? I sure hope not. Historically, Apple has had a tendency to tell developers about their grandiose technology plans and warn developers away from pursuing their own dreams because the "soon to be released" Apple technology will make obsolete those of the developer. Then, often enough, Apple's technology doesn't arrive until years later, if at all.

I don't mean to say that Rhapsody won't arrive, or even will be late. Apple has an ambitious development schedule, but they have dropped almost everything else to focus exclusively on the release of Rhapsody. This is probably a good thing given how Apple's future rides almost exclusively on the success of Rhapsody.

However, Apple's technology marketing has typically been about two years ahead of the technologies. That is, Apple convinces developers to adopt the new technologies a year or two too soon. This often leaves the developer struggling to meet cash flow requirements while Apple fixes bugs and the market evaluates the technology.

Apple's Official OS Schedule

Let's look at the Rhapsody schedule for a moment. Sometime this summer Apple will release to developers the first developer version of the new OS, but it likely will not be much more than NEXTSTEP on PowerPC. Early in 1998 Apple will offer the Premier Release of Rhapsody to all who want it. This release is expected to include all the planned features of the new OS, including a new Macintosh look and feel, but it will have only limited support for MacOS-based applications (the compatibility box). Late in 1998 Apple promises to release the Unified version of Rhapsody. This is supposed to be the finished product, and is likely to be the first version of Rhapsody to ship with hardware. This is the first version intended to replace the MacOS.

Over that same eighteen months Apple will release two major MacOS upgrades. These versions are intended to integrate the best of Apple's existing technologies and clear the shelves of anything interesting Apple had planned and was reasonably far along for Copland. We all expect the first such release, Tempo, will include a threaded Finder with the Copland look and feel. I'm not sure what to expect from the later release. If we are lucky, we will also eventually see V-Twin and Meta Content Framework (MCF-HotSauce). These same technologies are planned to be rolled into the Rhapsody compatibility box by the Unified release. These are the versions of the OS Apple will ship with hardware, at least until the Unified Rhapsody is available.

Yes Officer, I'll Slow Down

Over the years I have seen a great many Macintosh developers get excited about new Apple technologies at least a year too soon for them to make a living from the new technologies. Now may be the time for us to slow down a little and look at the OS we are living with right now. We have many great technologies that will be with us for many years to come. Maybe we should focus on those technologies a bit longer while we evaluate our ideas for Rhapsody.

When the developer version of Rhapsody is released later this summer will be a good time for us developers to take a look and see what is possible with the new, modern OS. It is quite likely there will be countless bits of software released to the community within days and weeks of the developer release, but most of that is just our exploring the technology. The real market won't even begin to develop until several months after the Premiere release, and that is plenty of time to learn the new OS and develop some cool software for it. Let's be patient and make sure we continue to put bread on our table by satisfying the market already at our doorstep.

To help you prepare for Rhapsody, MacTech will continue to bring you coverage of as much Rhapsody technology as we can get out of Apple. You can also find very useful descriptions and tutorials on the NeXT technologies from NeXT's web site at <http://www.next.com/>.

To help you take advantage of Macintosh technologies already available and soon to be released, we will also include coverage of them, including CyberDog, OpenDoc, Game Sprockets, Speech, OpenTransport, V-Twin and MCF. Although Apple has stated these technologies are not dead, just in maintenance mode, maybe some cool applications of these technologies will convince Apple of the marketing need to port them to Rhapsody. They are compelling technologies, and it is Apple's development of compelling technologies, and our desire for them, that has kept Apple alive all these years. Let's not ignore them.

MacHack on the Horizon

Many developers are looking for ways to quickly learn about Rhapsody. One of the best sources is likely to be June's MacHack. If Apple keeps to their OS schedule, there is a good chance the MacHack crew will have Rhapsody DR1 running in the Machine Room and several Apple engineers making presentations and looking for people to write cool hacks. Think of it as a huge coding kitchen. Check out <http://www.machack.com/> for more information about MacHack. MT

MORE DEVELOPERS PROTECT.

MacHASP Packs More Into Less.

Based on state-of-the-art ASIC technology, MacHASP packs the industry's most advanced security, compatibility and flexibility into a compact and end-user-friendly dongle.

Grow With Aladdin!

The fastest growing company in the industry, with over 4 million keys sold to 20 thousand developers worldwide, Aladdin is setting the standard for software security today.

NSTL Study Rates HASP No.1!

A recent test conducted by the National Software Testing Labs[†], the world's foremost independent lab, compared the flagship PC products of leading software protection vendors. The result? HASP was rated the clear overall winner - and number one in all the major comparison categories. For a full copy of the NSTL report, contact your local HASP distributor.

MacHASP® PROTECTS MORE.

Mac™OS

These days, more and more developers are choosing to protect their software against piracy. They're protecting more products, on more platforms, with better protection – and selling more as a result.

And more of these developers are protecting with MacHASP. Why? Because MacHASP offers more security, more reliability and more features than any other product on the market. Only MacHASP offers capabilities such as network support, anti-debugging envelope protection, and secure remote activation and updating.

MacHASP supports the most advanced platforms, including all versions of MacOS and Power Mac – as well as AppleTalk networks. And because Aladdin is a licensed Apple Partner, MacHASP guarantees full, transparent compatibility with the ADB standard.

To learn more about how you can protect better – and sell more – call now to order your MacHASP Developer's Kit.

The MacHASP Developer's Kit contains everything you need to protect your software today!

1-800-223-4277

www.aks.com

ALADDIN™

The Professional's Choice

North America	**Aladdin Knowledge Systems Inc.** Tel: (800) 223 4277, 212-564 5678, Fax: 212-564 3377, E-mail: hasp.sales@us.aks.com
Int'l Office	**Aladdin Knowledge Systems Ltd.** Tel: +972-3-636 2222, Fax: +972-3-537 5796, E-mail: hasp.sales@aks.com
Germany	**FAST Software Security AG** Tel: +49 89 89 42 21-37, Fax: +49 89 89 42 21-40, E-mail: info@fast-ag.de
United Kingdom	**Aladdin Knowledge Systems UK Ltd.** Tel: +44 1753-622266, Fax: +44 1753-622262, E-mail: sales@aldn.co.uk
Japan	**Aladdin Japan Co., Ltd.** Tel: +81 426-60 7191, Fax: +81 426-60 7194, E-mail: sales@aladdin.co.jp
Benelux	**Aladdin Software Security Benelux B.V.** Tel: +31 24-641 9777, Fax: +31 24-645 1981, E-mail: 100526.1356@compuserve.com

■ **Aladdin Russia** 095 9230588 ■ **Australia** Conlab 03 98985685 ■ **Chile** Micrologica 02 7350041 ■ **China** Shanghai URI 021 64377828 ■ **Czech** Atlas 02 766085 ■ **Denmark** Berendsen 039 577316 ■ **Egypt** Zeineldein 02 3604632 ■ **Finland** ID-Systems 0 8703520 ■ **France** 1 40859885 ■ **Greece** Unibrain 01 6756320 ■ **Hong Kong** Hastings 02 5484629 ■ **India** Solution 011 2148254 ■ **Italy** Partner Data 02 26147380 ■ **Korea** Dae-A 02 8484481 ■ **Mexico** SiSoft 91800 55283 ■ **New Zealand** Training 04 5666014 ■ **Poland** Systherm 061 480273 ■ **Portugal** Futurmatica 01 4116269 ■ **Romania** Ro Interactive 064 140283 ■ **Singapore** ITR 065 5666788 ■ **South Africa** D Le Roux 011 8864704 ■ **Spain** PC Hardware 03 4493193 ■ **Switzerland** Opag 061 7169222 ■ **Taiwan** Ieco 02 5559676 ■ **Turkey** Mikrobeta 0312 4670635 ■ **Yugoslavia** Asys 021 623920

by Dave Mark, ©1997, All Rights Reserved. http://www.spiderworks.com

More Objective-C

Last month, we took our first taste of Objective-C. We learned that Objective-C sources use the extension ".m" for main source files and ".h" for header files. The type "id" is used to declare a generic pointer which allows us to delay type binding decisions. We learned that most objects are derived from the Root class Object. The Object class features variables and methods inherited by all other classes. One of these variables is "isa" which specifies the class to which an object belongs.

We saw the form for a class interface:

```
@interface MyClass : MySuperClass
{
  instance variable declarations
}
method declarations
@end
```

We also learned about Objective-C's funky form of method declaration:

```
- (int)getX:(int)x andY:(int)y;
```

Where the leading minus sign marks the function as an instance method — as opposed to a "+", which marks a method as a class method, like a C++ static method. The first "int" is the return type, and the other "int"s are the parameter types. The name of the method above is "getX:andY:" The default type is id, so if you leave out any of the types (including the return type) the type is set to id.

We also learned about the #import compiler directive, used to avoid multiple inclusion of a .h file:

```
#import "Object.h"
```

Alternatively, you can avoid all the extra junk you get when you import a classes' header file by using the @class directive to let the compiler know that a reference is to a legal class:

```
@class Object;
```

Finally, we saw the form used for the actual implementation of a class:

```
#import "MyClass.h"

@implementation MyClass
method definitions
@end
```

Message Receivers and Message Syntax

Now it's on to new material. Once you declare your class and use that declaration to define an object, it's time to bring that object to life by sending it a message. In C++, you call an object's member function using the object itself to make the call:

```
myObjectPtr->MemberFunction();
```

In Objective-C, a message is said to be sent to a receiver using the following syntax:

```
[receiver message];
```

The receiver is an object and the message is a method, along with its associated parameters. The message is sent (and the appropriate method selected) at runtime. For example, suppose the class named MyClass included the following method:

```
- (int)getX:(int)x andY:(int)y;
```

Just as a reminder, this method is called getX:andY:, takes two parameters (both of them ints), and returns an int. Here's a line of code that sends a message asking a MyClass object to perform the getX:andY: method:

```
myNum = [myObj getX:27:50];
```

In this case, the object myObj (we'll see how to allocate an object in a second), presumably of class MyClass, gets sent the getX:andY: message, along with the parameters 27 and 50.

Mac, go out and touch the world for only $199!

ADB I/O lets your customers' Macs control things, it lets them feel.
ADB I/O lets the Mac be part of the physical world.

Thousands of Uses

Science, Multimedia, Children's Museums, Home Automation, Theatre Stages, Industrial Testing, Medical Research, Bonsai Watering, Robotics, Weather Stations— anything that can be electronically measured or controlled can use the ADB I/O.

No Serial Ports Occupied

ADB I/O uses the Apple Desktop Bus to communicate inputs and outputs to and from your Macintosh. (Maximum polling frequency is 90 Hz.) No external power supply is needed.

Eight I/O Channels Provided

Four relays for output. Four channels for Digital In, Digital Out or 8-bit Analog In.

Extensive Software Support

With ADB I/O and nearly any environment,* it is easy to build customized applications for your control and data acquisition needs.
For more info, visit us at
www.bzzzzzz.com.

BEEHIVE
TECHNOLOGIES INC.
The High-Tech Cottage Industry Cooperative On the Internet

*ADB I/O supports three language environments, two scripting environments, multimedia development environments, and other specific application environments with more on the way. Visit our home page to find out more and to download the complete manual as well as all supporting software at: **http://www.bzzzzzz.com.**
(818) 304-0664 voice. e-mail: contact@mail.bzzzzzz.com (818) 568-1530 fax. © 1997 BeeHive Technologies, Inc. All rights reserved.

ALLOCATING AN OBJECT

To allocate a new object, you'll send an alloc message to the class whose object you want to create. Here's an example:

```
id myObj = [[MyClass alloc] init];
```

You'll use this line of code again and again, with a few alterations of course. We start off by defining an id to hold the reference to the allocated object. The right side of the "=" operator is a message expression embedded in another method expression, much as you might embed a function call inside a second function call (printf("%d", GetMyValue()); for example).

In this case, we are first sending the alloc message to MyClass. The alloc method is a factory method (it is declared with a leading "+" sign instead of a leading "-"), similar to a static member function in C++. It is inherited from the Object class. The alloc method allocates the memory for a single object of the specified class and returns a pointer to the object.

That object is then sent the init message, which causes that object's init method to be performed. A classes' init method should return a pointer to the object (otherwise this code wouldn't work). The object pointer returned by this nested pair of message expressions is assigned to the object pointer myObj.

As you would expect, Objective-C has an analogy to the C++ keyword "this". To refer to the current object, use the term "self". When you write your init method (to initialize your newly alloc'ed object), you'll end it by returning self:

```
return self;
```

OUR FIRST OBJECTIVE-C PROGRAM

Let's bring all these concepts together with a simple example. We'll create a Number class that features a single variable, an int to hold the Number's current value. We'll add methods for initialization, one to square the Number's current value, and one to print out the current value. While this example may seem trivial, it will act as a nice syntax reference as you build your own classes.

By the time you read this, Metrowerks should have delivered their first Objective-C tools (scheduled for release as part of May's CW12). Since I don't have those tools in hand yet, I had to look at alternative Objective-C environments. As I mentioned last month, Tenon makes a Mac environment called CodeBuilder which supports the gcc Objective-C compiler. CodeBuilder supports an X environment called AfterStep, along with a more conventional UNIX terminal environment running either C Shell, T-C Shell (an improved C Shell), the traditional Bourne Shell (less powerful, but uses less memory), and the Bourne Again Shell (bash, Bourne Shell with improvements). All this rides on top of a Berkeley 4.4BSD UNIX system.

If you are into UNIX, CodeBuilder is definitely a lot of fun to play with. On the other hand, I find myself aching for the CodeWarrior IDE and editor to tie my projects together. Though I must admit that I did enjoy using vi to edit all my source code. Amazingly, I remembered all those cryptic vi commands that made vi such a pleasure/pain to use. Ah, well, enough reminiscing — on to the code...

The first thing I did was create a new folder to hold the source code (if you are using CodeBuilder, the UNIX command "mkdir dirName" creates a new directory, "cd new dir" changes directories, "mv oldname newname" changes file or directory names, "rmdir dir" deletes an empty directory, "vi filename" invokes the vi editor and "man command" brings up an online manual for the specified command. If you are new to UNIX or to vi, I would definitely try "man vi" to get a sense of the editor before you get into it).

Inside the new folder, I created three source files: Number.m, Number.h, and main.m. Here's the source for Number.m:

```
#import "Number.h"
//#include "Number.h"

@implementation Number

- init:(int)startValue
{
  [super init];

  value = startValue;

  return self;
}

- squareSelf
{
  value *= value;

  return self;
}

- print
{
  printf( "Number value: %d\n", value );

  return self;
}

@end
```

The release of CodeBuilder I had did not support #import (at least not in the usual fashion). If your environment does not support #import, comment out that line and uncomment the #include. Remember, if you use #include, you'll need to also make a corresponding change to the header file (we'll get to it next).

Number.m contains the implementation of the Number class. All three methods are instance methods and are declared using the leading "-". Notice that the init method takes a single parameter, startValue, an int used to set the value of the instance variable named value. Value is declared in Number.h.

The init method starts off by sending an init method to its superclass (in this case, the Object class). Doing this gives your superclass a chance to initialize its superclasses and itself, a real good idea. After setting value to startValue, init returns a pointer to the newly initialized object (remember, self is like this in C++).

squareSelf and print are fairly straightforward. Note that both return self, though they don't necessarily need to, since their return value is ignored by main. Also, you can see that the C standard library function printf() is available in Objective-C. As Rhapsody starts to kick in, I'll start making use of Mac interface calls instead of using console i/o but, for now, we need to make do with what we have.

Here's the source for Number.h:

```
//#ifndef _Number_h_
//#define _Number_h_

#import <objc/Object.h>
//#include <objc/Object.h>

@interface Number : Object
{
  int value;
}

- init:(int)startValue;
- squareSelf;
- print;

@end
//#endif
```

Once again, if your environment doesn't support #import, uncomment all the commented code above and comment out the #import line. Note the declaration of the instance variable named value along with the declaration of the three methods init, squareSelf, and print. Note that all three return an id, since no return value is declared. That's why the:

```
return self;
```

at the end of each method makes sense.

Here's the source for main.m:

```
#include "Number.h"

void main()
{
  id number = [[Number alloc] init:27];

  [number print];

  [number squareSelf];
  [number print];

  [number free];
}
```

main() is just a sequence of method expressions. First, we declare number to be an object pointer, then we allocate and initialize a new Number object and assign the object's pointer (returned by init) to number. Note that the value 27 was passed in as a parameter to init. Got it?

The rest of the program is cake. Send number the print message. Here's the line of output that appears in the console window:

```
Number value: 27
```

Next, send number the squareSelf message. This causes the number object to square the value in value. Before you reach for your calculators, 27 * 27 = 729. Really.

To verify this, we pass another print message on to number. Here's the line that appears in the console window:

```
Number value: 729
```

Finally, we send the free message to number. Since the Number class does not override the Object classes' free method, the Object classes' free method is the one that ends up getting called free. The Object version of free deallocates all memory allocated for the object by alloc. Note that the Object version of free does not follow any pointers in your object to free any secondary memory. If you do allocate additional memory in your init method (don't override alloc), you'll want to override free and deallocate the additional memory.

If you do override free, be sure to send the free message to super:

```
[super free];
```

at the end of your free method to give super classes a chance to clean up after themselves.

By the way, if you are using CodeBuilder, you might want to take advantage of the UNIX tool called Make. Create a file called Makefile in the same directory as your sources, and type in these two lines:

```
Number:  main.m Number.m Number.h
  gcc -o Number main.m Number.m -lobjc
```

Now you can recompile your program by just typing the command Make.

TILL NEXT MONTH...

I am really enjoying the opportunity to mess with Objective-C. Though the syntax threw me at first, it didn't take me long before I was thinking in it. This is not a hard language. Just a bit weird! I'm definitely looking forward to getting Rhapsody up and running on my machine so I can experiment with true dynamic binding. Excellent!

Just a heads up — I have gotten a lot of email asking me to update my How to Get Started With Mac Programming article. So much has changed in the Mac universe (Rhapsody, the Internet, Objective-C, Java, etc.) that the old recommendations just don't hold water anymore. Within the next couple of months, I'm going to interrupt the Objective-C series (just for one month!) to run a new version of that article. For those of you who are already way Mac programmers, please bear with me. Till then, it's back to UNIX, vi, and Objective-C... MT

Want to suggest an article for the magazine? Send your suggestion to <mailto:editorial@mactech.com>

PLATFORMS: APPLE SYSTEM 7 • APPLE A/UX • WINDOWS NT • WINDOWS 95 • OS/2 • NLM • DOS • LYNX • SCO • SYSTEM MANAGER • SUN SPARC-SOLARIS • IBM RS/6000 • HP9000 • RS/6000 • SUN O/S 4.X • NT (ALPHA) • MIPS ABI (SGI) • AT&T UNIX • BANYAN • INTERACTIVE UNIX • LINUX • MOTOROLA 88OPEN • QNX •

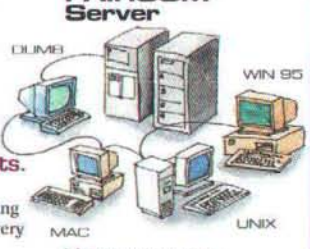
Q: What does it take to deploy a superior client/server application?
A: *A SUPERIOR SERVER*

***START* with the most advanced client-side SDK on the market: c-tree® Plus at $895.**
- **Complete "C" Source code**
- ROYALTY FREE (Client Side)
- Multiple supported protocols
- Fast, portable, reliable
- Powerful features like transaction processing
- Win95, NT, and Windows 3.1 ready

***RESULT?* A solid, economical, easily deployable product that fits your needs.**
- Portable
- Scalable
- Exceptional Performance
- Flexible
- Easy Server distribution
- Convenient OEM terms

FAIRCOM® Server

***ADD* a strong, multi-platform, industrial-strength Server that supports.**
- File mirroring
- Heterogeneous networking
- Automatic disaster recovery
- Multi-threaded design
- Best price/performance available: from $445- $3745



Heterogeneous TCP/IP Network

You can't find a better client SDK with these features! Over sixteen years of proven reliability and performance. No one else supports over 30 platforms in this price range!

c-tree Plus®
- Complete C Source
- Single/Multi User
- Client/Server (optional)
- Full ISAM functionality
- No Royalties
- Transaction Processing
- Fixed/Variable Length Records
- High Speed Data/Index Caching
- Batch Operations
- File Mirroring
- Multiple Contexts
- Unsurpassed Portability

FairCom® Server
- Client/Server Model
- Transaction Processing
- Requires <2MB RAM
- Online Backup
- Disaster Recovery
- Rollback - Forward
- Anti-Deadlock Resolution
- Client-side "C" Source
- Multi-threading
- Heterogeneous networking
- File Mirroring
- OEM/Source Available

FOR YOUR NEXT PROJECT CALL FAIRCOM: YOU CAN'T FIND A BETTER HETEROGENEOUS CLIENT/SERVER SOLUTION!

Also inquire about these FairCom products:
d-tree™
r-tree®
ODBC Driver

FAIRCOM® CORPORATION

Since 1979

WWWeb Address: http://www.faircom.com/
800-234-8180

U.S.A. phone (573) 445-6833 fax (573) 445-9698
EUROPE phone (035) 773-464 fax (035) 773-806
JAPAN phone (0592) 29-7504 fax (0592) 24-9723

by David Neumann, System Engineer, NeXT

An Introduction to WebObjects

An overview of Apple's new high end application development & deployment vehicle for the Internet

This article gives a technical introduction to NeXT's WebObjects web application development platform. The content is primarily geared toward developers with at least some Internet-based application building experience.

First and foremost, WebObjects means server-based application development. It is not an authoring tool. When you layout a page with WebObjects, you are laying out your application's user interface.

WebObjects as a product ships with three major components: a cross-platform set of object-oriented frameworks, developer tools on top of those frameworks, and a deployment infrastructure for running apps built on those frameworks. This article will go into detail on all three components. First, we'll delve into "Logical Architecture" of the frameworks, and then "Physical Architecture" of deployment, and finally tools & custom code used to wire the functionality together.

Although WebObjects is full of highly extensible objects ideal for heavy-duty customization, WebObjects also includes tools that make it a first-class "RAD" (rapid application development) environment for the web. For instance, WebObjects includes drag-&-drop access to industrial-strength relational databases and offers a Database Wizard for creating complete database-aware applications (or parts of applications) with a few mouse clicks.

Note that a developer may elect to write a WebObjects application in either Java, Objective-C, or WebScript. Virtually all of NeXT's examples ship with both Java and Objective-C/WebScript equivalents. However, the developer does not have to stick to a single language, all three may be used simultaneously. For example, NeXT encourages the use of a compiled language like Java or Objective-C for business logic (for example, rules, policy, and vertical-oriented computations) and WebScript for "application" logic (such as, when you punch this button, select this field and jump to page XYZ, etc.).

WebObjects as a product ships with three major components: a cross-platform set of object-oriented frameworks, developer tools on top of those frameworks, and deployment infrastructure for running apps built on those frameworks.

LOGICAL ARCHITECTURE

Below is a look at the logical architecture of the WebObjects family of frameworks. Frameworks are the programming building blocks of the largest granularity. These frameworks include: Foundation (the non-GUI aspect of the OpenStep spec), Enterprise

David Neumann has been a Systems Engineer with NeXT Software for nearly two years. Using WebObjects, he has assisted many customers with web application prototypes and production online applications including that of a major PC computer vendor. He has also written WebObjects software (available as an example on CD to WebObjects 3.0 customers) enabling the management, monitoring, and administration of distributed, large scale WebObjects applications.

Increase Your Software Sales

WIBU-KEY™ Copy Protection provides the most secure and flexible software protection available.

WIBU-KEY has been protecting software around the world since 1989. WIBU-KEY combines the most sophisticated encryption technology with the highest grade hardware components to give you peace of mind that your software won't be used illegally. You've worked hard to create your Mac application. Don't settle for anything less than the best copy protection system - WIBU-KEY.

- Hot plug 'n play ADB design
- Network support built-in
- Protect multiple applications or program modules
- Secure Remote Programming
- Consistent API for Mac and PC development
- User-friendly setup and diagnostic tools for your customer

800-986-6578

www.griftech.com

GRIFFIN TECHNOLOGIES™ 1617 St. Andrews Drive • Lawrence, KS 66047 Tel: 913-832-2070 • Fax: 913-832-8787 Email: sales@griftech.com

WIBU™ SYSTEMS Rueppurrer Strasse 54 • 76137 Karlsruhe • Germany Tel: +49-721-931720 • Fax: +49-721-9317222 Email: info@wibu.de

Belgium, Lux. Compusec Tel: +32-2-6450944 Fax: +32-2-6464266 ■ **Croatia** Aries D.o.o. Tel: +385-1-222752 Fax: +385-1-2326535 ■ **Estonia** Lansoft Ltd. Tel: +372-2-444760 Fax: +372-2-682760 ■ **Japan** Visual Networks Co. Ltd. Tel: +81-3-340-57801 Fax: +81-3-340-57818 ■ **Netherlands** Compusec Tel: +31-53-5740223 Fax: +31-53-5726822

Objects Access (bidirectional RDBMS-to-object mapping), Enterprise Objects Control (persistent object transaction management), and WebObjects (web presentation & deployment).

Note that you may have an application split into separate services or tiers that use all of the frameworks and/or only Foundation and/or only Foundation+Enterprise Objects Access & Control. It is also possible to write a WebObjects application that only uses Foundation and WebObjects (no object/relational datasource). See the side-bar "WebObject Class Functional Groupings" for details about classes available in the WebObjects Frameworks.

All applications must include the Foundation Framework. It provides basic services needed by any program. It is also the layer of objects that provides operating system independence. In other words, you should never have to make a direct system call again for most applications when you have the luxury of using Foundation. Foundation provides operating system independence across the upcoming Rhapsody OS for Mac, Windows NT, Solaris, HP-UX, and of course NeXT's OPENSTEP/Mach operating system.

Visit MacTech Magazine's Web site!
http://www.mactech.com

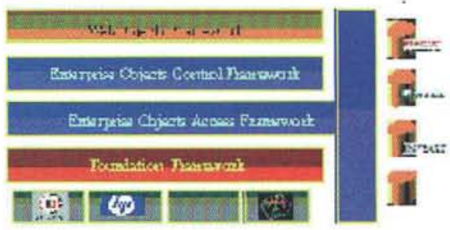

Enterprise Objects Control Framework
Enterprise Objects Access Framework
Foundation Framework

Figure 1. *WebObjects Frameworks. Foundation provides operating system independence, Enterprise Objects Access provides RDBMS independence for object persistence, Enterprise Objects Control provides object persistence transaction management, and WebObjects provides objects useful for web based presentation and deployment. Note that actual linking dependencies are different than implied in the figure.*

The Enterprise Objects Frameworks (Control & Access) also known collectively as "EOF" are extremely feature rich. Truly doing EOF technology justice would require an entire article unto itself and so unfortunately this discussion can only skim the surface. Consider that EOF is NeXT's third generation solution for database aware applications. The sidebar on "WebObjects Class Functional Groupings" should give the reader at least a flavor for what's available in the technology.

It's EOF that facilitates the use of true business objects in WebObjects. EOF makes your application completely database and schema independent. With it, you can create "live" entity-relationship diagrams in a tool called "Enterprise Objects Modeler" (or just "EOModeler") that ships with WebObjects. EOModeler will emit new schema and (perhaps more importantly) let you reverse engineer an existing relational database into a default object mapping — that you can then enhance without affecting the underlying schema. You can write business logic in the same language you write your application code and take advantage of both composition and inheritance relationships in that business logic. This is highly different from "normal" approaches to database driven applications: 1) forcing policy into a proprietary, database-specific stored procedure language (not object-oriented) or 2) scattering business logic across applications in event handlers. Furthermore, EOF makes it trivial to build applications that integrate resources from several databases — not just one. Add-in technology from third parties allows you to extend this integration to mainframe applications via 3270, 5250, and ASCII data streams.

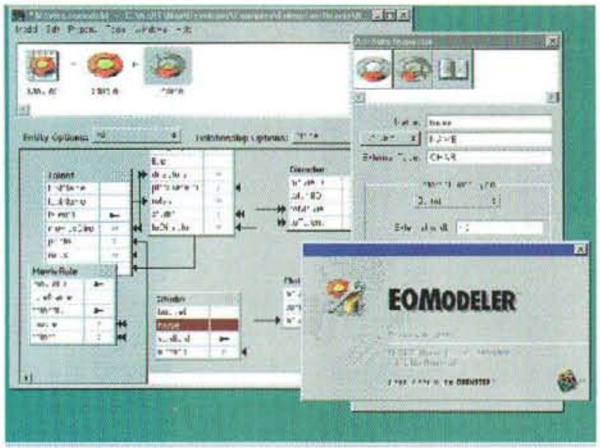


***Figure 2.** EOModeler application. This tool ships with WebObjects to let developers graphically create "live" entity relationship model of their business objects. The model can be targeted to any datasource known to WebObjects including Oracle, Informix, Sybase, and ODBC. Note how the attribute (ìnameî) on the selected entity (ìStudioî) is abstracted from its external database name and type. If the datasource changes, only the external information need change insulating application code from the database implementation.*

The fundamental difference between WebObjects database access compared to traditional data access methods is this; when you fetch a "row" from a database, you do not get naked data, you get an object with behavior, an object that knows about its relationships to other objects. You get an object that is watched, uniqued, deleted, and updated from within a document context (kind of like a super transaction) with automatic support of undo, redo, revert, and efficient change flushing to the underlying database or databases. You specify the kind of object you get for a row (for example, its Class) using EOModeler.

Since the business objects you create with EOF don't care about the underlying database or how their values are presented in user interfaces, they may be re-used over and over in any number of different applications (web, Windows, or Macintosh Rhapsody) and indeed may be maintained by a crew of developers only loosely coupled to those building specific apps.

PHYSICAL ARCHITECTURE

Now that we know what kinds of building blocks are available from a developer's perspective, how does that logical architecture get laid out physically into a running application? **Figure 3** shows the tiers involved in a typical deployed WebObjects application.

Unlike traditional modes of Internet programming such as Perl scrips, a WebObjects application does not "disappear" between page processing/generation transactions. It stays up and maintains a persistent database connection. The task of state management is completely subsumed by the WebObjects framework. State management is also abstracted to the point that alternative state management policies may be implemented with little or no impact on existing code (such as, state in the process, state in the file system, state in the page, state in the someplace-not-invented-yet, etc.).

With WebObjects, browsers interact with HTTP servers in their normal way taking advantage of any technologies that enhance this browser-to-web server link. For example, secure socket layer communication protocols in Netscape and Microsoft browser/server products.

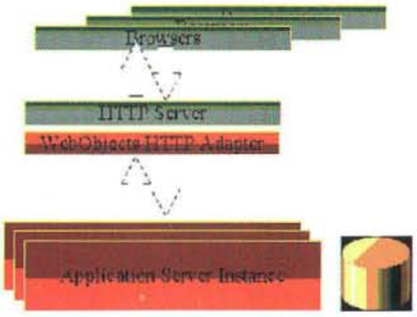


***Figure 3.** WebObjects Deployment. Browsers communicate with HTTP servers which communicate with the WebObjects web server adaptor. The WebObjects adaptor load balances across one or more application server processes (also called instances) running on one or more machines. Once running, WebObjects application instances do not go away between user requests; they maintain themselves, their sessions' state for users, and their database connections. They are efficient, fast, and by definition, redundant.*

It's the job of the HTTP server adaptor, as indicated in **Figure 3**, to communicate with a given HTTP server and forward requests to one or more application "instances" — an instance is a separate copy of a given application process. A WebObjects app serving a few users may have only one instance. A large application may have tens or hundreds of instances running on wide

sesame

door

mind

season

architecture

for business

possibilities

horizons

house

now

platform

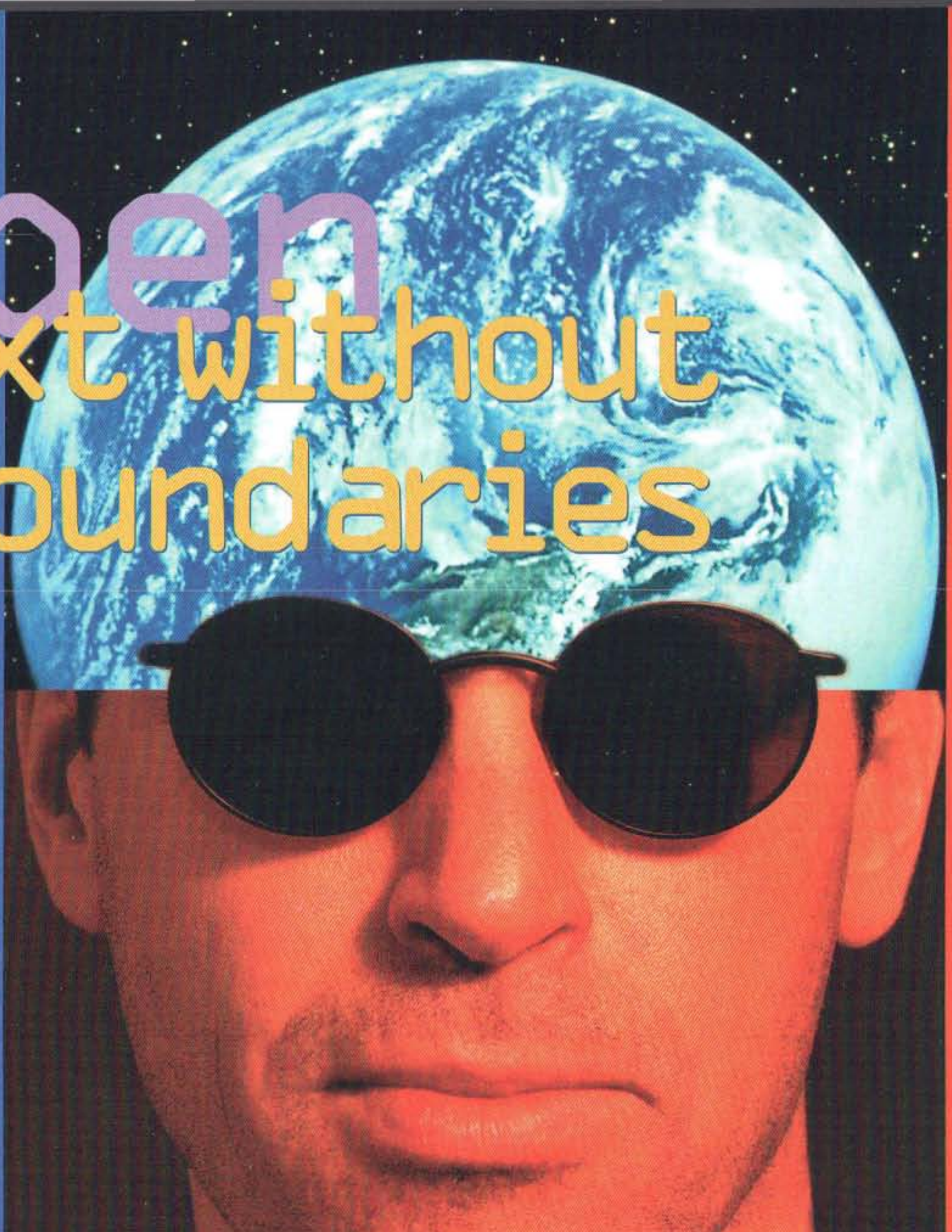
window

road

sky

borders

up



WITHOUT US THEY WOULDN'T EXIST (OR AT LEAST THEY'D BE MUCH MORE FRUSTRATED.)

America Online

Apple Computer

Broderbund Software

Canon

Claris

CompuServe

Dataware

Digital Equipment

Fujitsu

GE Information Systems

ichat

Macromedia

National Geographic

Now Software

QualComm

Seagate Software

3M

US West

Xerox

CONSIDER the possibilities. They're wide open. Simply put, OpenPaige™ can do anything you want to the text of your software application.

Across platforms, across architectures, even UNIX™. OpenPaige is a feature-rich text environment. Every aspect of text and layout formatting, editing and display is possible. Stylized text. Scaling. Container and non-rectangular shapes. Style sheet support. Cut, copy and paste. Embedded pictures, objects, buttons and even calls to other applications are at your fingertips.

Don't waste time struggling with your text. We've already done the work for you. Drop in OpenPaige and go. Use as much or as little of it as you wish. Available as object code or as source code, it has an elegantly small footprint. Yet it will empower you to open doors, boundaries, and minds with your software application.

But don't take our word for it. Check out our extensive and diverse client list. They've all discovered how the robust and flexible power of OpenPaige can open up entire new worlds of possibilities.

Open Paige™

Datapak

For additional information, check out our web site at www.datapak.com/~datapak/ or call us at 1-800-327-6703.

OpenGL® for the **Macintosh**™

The Industry Standard for 3D Graphics

- Port your code from other platforms with ease!
- Workstation Class Performance on your Mac
- Complete on-line documentation and support
- All common support libraries provided
- Multi-processor capable
- UNIX or MKLinux versions
- Works with most compilers, including CodeWarrior ®
- Callable from C/ C++, Ada and FORTRAN
- Full Technical Support
- Custom Libraries available

Conix Graphics

Download free demos at:
www.conix3d.com

Let your imagination fly!

Sales	Technical	FAX
800.577.5505	817.467.0461	817.467.9452

sales@conix3d.com

All trademarks remain the property of their respective owners.

one or more machines. If an application has more than one instance, the WebObjects adaptor is essentially acting as load balancing agent. If an instance fails, it is only affected — other instances and/or the site's web server is unaffected.

WebObjects HTTP Server adaptors come in three flavors: an NSAPI plug-in, an ISAPI DLL, and a very low overhead CGI program. These adaptors are written in pure C and all of their source is included with WebObjects. This was provided as a portability & integration strategy. Consider that WebObjects applications may reside on the forthcoming Rhapsody, Solaris, Windows NT, etc., but not all operating systems. To facilitate plugging in a cluster of WebObjects Application server boxes into an existing infrastructure of, say, SGI HTTP servers, the developer may recompile the desired web server adaptor flavor for their target operating system.

The adaptor will forward requests over the network as easily as it will forward requests to applications running on the same box as the HTTP server. In fact, from a load sharing perspective, it is ideal for the HTTP server and Application servers to reside on separate boxes. This way, WebObjects can generate pages on the fly without competing with the HTTP server's CPU(s) that may already be quite busy vending out the web site's "static" HTML pages.

Since WebObjects applications are server based, database access happens behind the firewall. Browsers need never make direct connections to a database server. Database connections are therefore highly secure (only accessible via actual application API), and conserved (that is, you never have more than one connection per instance regardless of the number of users supported — unless this is specifically something the developer desires).

Any browser can be a WebObjects client since only standard HTML need be generated on the fly by the WebObjects server. If the developer desires a more advanced look & feel, then other content types may also be generated such as Java applets. WebObjects essentially makes a distinction between client Java and server Java. Client Java is used to enhance the user interface (e.g. a Calendar applet, a simple field validation applet, etc.). Server Java (and/or server Objective-C/WebScript) is used to customize the WebObjects frameworks to the business logic and behavior requirements of your application (for example, a Customer object, a stat-gathering Advertisement component, etc.).

In addition to HTML and HTML embedded with Java applets (and/or any other client side technology driven by HTML content, e.g. ActiveX, RealAudio, etc.), WebObjects 3.1 includes one new content generation feature: support for dynamically filling in "blanks" in PDF documents using Adobe's new FDF protocol. This capability lets WebObjects applications create visually appealing and printable reports, forms, and catalog pages dynamically.

PROGRAMMING A WEBOBJECTS APPLICATION

While all operating systems supported by WebObjects are development platforms in the sense that they all support building WebObjects application executables, Windows NT, OPENSTEP/Mach, and Rhapsody are considered the "preferred" development platforms because only on these platforms are all of the graphical tools supported. These tools include WebObjects Builder (a web-oriented user interface layout tool with drag & drop database access), EOModeler (the object to RDBMS mapping tool for abstracting your datasources), and Project Builder (a project management, compiling, & debugging environment). If your application uses only WebScript, you will not need the management services of Project Builder.

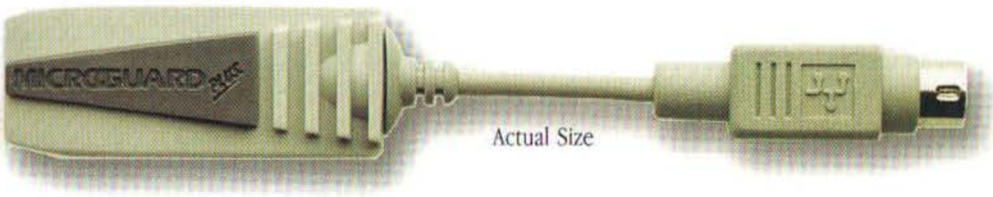
All resources in a WebObjects application are placed underneath the application's "wrapper" which has a "woa" extension. For example the DodgeDemo database-fed catalog application that ships with WebObjects is actually a directory with the name "DodgeDemo.woa" filled with resources. These resources include components, pages, images, scripts, source files, etc.

You use WebObjects Builder to create your application's "components" which are assembled into the pages the user sees. Some components represent a page in total. Some represent a part of a page (like a footer). Components can be nested within each other. Action handlers representing a component's behavior may be expressed in WebScript, Objective-C, or Java (or no code at all if all the behavior is in components prebuilt by you or NeXT). **Figure 4** shows the WebObjects Builder while a standard reusable component (the WebObjects active image logo) is being dragged & dropped on the Main component of the HelloWorldJava application.

You already know that...

MicroGuard Copy Protection is -

UNBEATABLE



So... Here's how you can reach us:

www.micromacro.com

International
Micro Macro Technologies, Ltd.
3 Hashikma St.
P.O.Box 11516, Azur 58001
Israel

Tel: (972-3) 558-2345
Fax: (972-3) 558-2344
E-mail: info@micromacro.com

USA
MicroGuard
631 South Pontiac St.
Denver CO. 80224
USA

Tel: (303) 320-1628
Fax: (303) 320-1599
E-mail: usa@micromacro.com

Mac™OS

MICROSOFT® WINDOWS™ COMPATIBLE

MICROGUARD PLUS.™

Charter member Pirate Busters

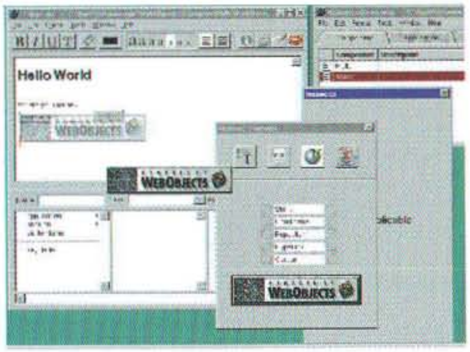


Figure 4. *WebObjects Builder user interface layout. Here a prebuilt component (the WebObjects active image logo) is being dragged and dropped onto the HelloWorldJava application's "Main" page HTML "canvas."*

Note that the upper pane shows the component's visual layout while the lower pane shows all of the methods and objects known to the component. Methods are listed below the line in the left-most browser column, variables above it. We see that this component has defined an action method called "sayHello" and an instance variable called visitorName. Notice that all components have access to any variables and methods implemented on the Session and Application via their respective entries in this browser.

Using WebObjects Builder can be summed up as the act of associating methods and variables in the lower pane with the properties of custom and NeXT components placed on the page canvas in the top pane and repeating this process for however many of your custom components make up the application.

When you add code to a component (regardless of the language used), you are subclassing the WebObjects class WOComponent (or your own subclass of WOComponent). A component definition is stored in a folder with a "wo" extension. Every component includes three basic files: an HTML template (which defines pure layout), a code file (defining pure behavior), and a "declaration" file that maps the two together. Using this approach, code can change without impacting layout and format/look can change without impacting code. It also makes it straightforward to delegate HTML look issues to a third party that may have special expertise in page layout. Developers can import the HTML at a later date for the final application release (or change the look dynamically while the application is running). When you edit a component in WebObjects Builder, it manages all of the component's resource files transparently.

All applications include a WOSession and WOApplication object whose behavior you can customize with code in either WebScript, Objective-C, or Java. Any variables you declare in your WOApplication subclass represent Application scope (for example an array of URLs pointing to third party sites). Any variables declared in your WOSession subclass represent Session scope (for example, a user's shopping cart or active selection). Any variables declared in your WOComponent subclasses represent component scope unique to each component (for example, the disclaimer string in a footer component). Components tend to last for only a single transaction, sessions last for many transactions, and an Application contains many sessions.

At your discretion, you may elect to override several important methods that allow customization to the event handling process. These include, but are not limited to: "init" — called once and before anything else on object creation, "awake" — called before action handling on every request, "sleep" — called at the end of an event on every request, and "dealloc" — called right before the object is destroyed. All four methods may be implemented at all three scope levels (i.e. in your Application, Session, and Component subclasses).

Below is source code for the HelloWorld example's "Main" page. The Main page is the very first page that gets created when a new session is created. This example is implemented in Java. You can access a component's code via the code inspector as indicated in the **Figure 5**.

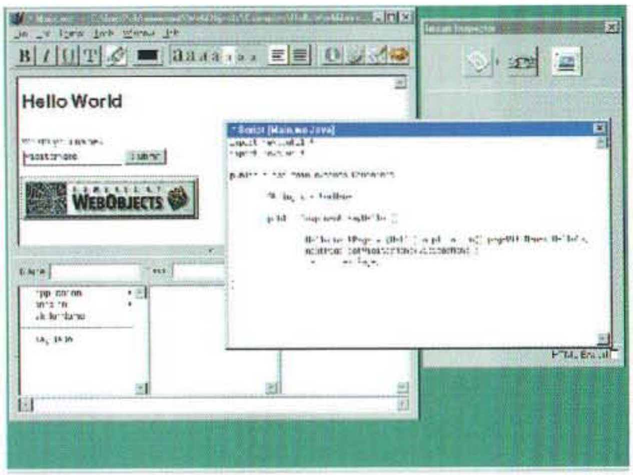


Figure 5. *WebObjects Builder showing code inspector with Main component's sayHello event handler and visitorName instance variable defined inside.*

Listing 1: Main.java

A custom subclass of the WebObjects Java class "Component" (WOComponent in the Java name space). The NeXT Java frameworks are imported as next.util.* and next.wo.*. This subclass declares a Java String as an instance variable and a single action method.

```
import next.util.*;
import next.wo.*;

public class Main extends Component
{
  String visitorName;

  public Component sayHello()
  {
    Hello nextPage =

    (Hello)application().pageWithName("Hello");
    nextPage.setVisitorName(visitorName);
    return nextPage;
  }
}
```

We see that the class Main is a subclass of Component (the class name of WOComponent in the Java name space). This component is also the entire page. It declares a single page variable, visitorName, and a single event handler method, "sayHello()". The first line in the method invokes the method "pageWithName()" against the Application object. All WOComponents inherit a method "application()" that returns the application object for convenience. pageWithName() takes the name of the page instance you'd like to get a reference to as an argument. If it finds the page defined in your application wrapper, it returns it. Here we're taking the returned page object and assigning it to a variable "nextPage". In the next line, we invoke a method on nextPage "setVisitorName()". Finally, on the third line, we return nextPage which means this page will define the response for this transaction. If we had returned "self", the same page that handled the previous request would make up this response.

You may be wondering where the value for visitorName is getting set. In WebObjects Builder, HelloWorld's Main component has a single WOTextField object whose value property has been mapped to this component's visitorName instance variable. This means that whatever value it has on page generation will fill the textfield upon display, and any value the user enters will get automatically parsed and placed in this variable by the time your action method is invoked. In **Figure 6**, WebObjects Builder's element inspector is focussed on the textfield. Note that the textfield's value property is set to the component instance variable.

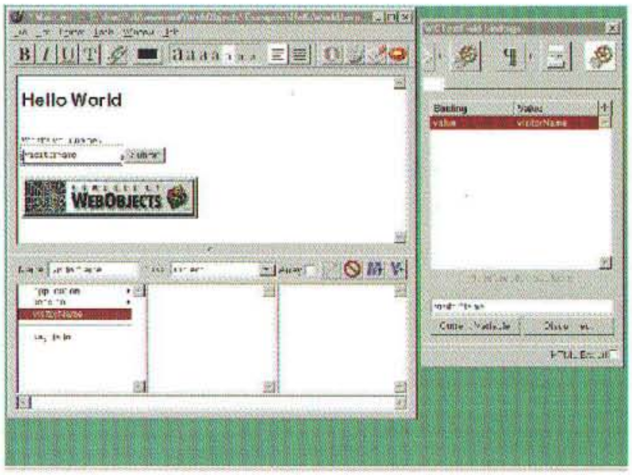


Figure 6. *WebObjects Builder showing WOTextField inspector with its value property mapped to the highlighted visitorName component instance variable.*

If we wanted the textfield to have an initial default, we could elect to set it in an awake method for the page as shown in Listing 2.

VOODOO

The ultimate version control tool for Macintosh

- Stand-alone tool with easy-to-use graphical user interface
- Suitable for all sorts of projects (software development, documentation, design, CAD, publishing, etc.)
- Simple and clear management of variants and revisions of entire projects (not only single files)
- Administration of users with hierarchical access rights
- Recording of the complete project history (who made which changes when and why)
- Efficient delta storage (95 % savings and more) for files of arbitrary type

NEW VERSION 1.8

- Smooth integration with Metrowerks CodeWarrior IDE
- Support of AppleEvents and AppleScript
- Comparison of files of different types (not only text files)
- Configurable local file locking (Finder flag or CKID resource)
- Improved handling of local files including the creation of folder structures
- Significant performance improvements in many places
- Essential parts PowerPC native

UNI SOFTWARE PLUS
A-4232 Hagenberg Austria
email: voodoo@unisoft.co.at --- http://www.unisoft.co.at/e/products/voodoo.html

Automated testing just got easier:
query & control the interface

$95.00
60-day MBG
Runtime: $25 ea.

"An excellent tool for regression testing!"

PREFAB **PLAYER 1.1** (version 1.0 shipped Aug. 1994)

Like a player piano, **Player mimics user input.**
Your scripts can choose from menus & pop-ups, select radio buttons, type text, determine the name of the frontmost window, the state of a checkbox, etc. ***For AppleScript and Frontier.***

Complete details and free 30-day trial version on our website.

Brian M. Criscuolo, DataViz

The simple way to handle complex text™

$95.00
60-day MBG

see website for discounts on pre-release versions

PREFAB **TEXT MACHINE** (version 1.0 to ship Q2 1997)

Give Frontier and AppleScript the power of Perl.

Search and replace **patterns** of text using simple keywords instead of cryptic grep syntax.
Cleanup and convert text for web publishing, e-mail management, DTP, database import, etc.

Version 1.0 will support AppleScript & Frontier,
Version 1.1 will add a dialog for OpenDoc.

Net Tip

420A Main Street
Acton, MA 01720
508-266-2885

PREFAB SOFTWARE, INC.™

player@prefab.com

www.prefab.com

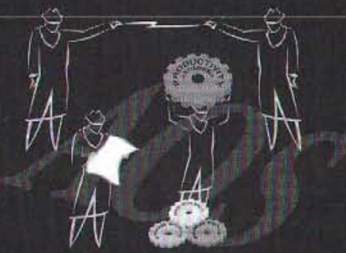
SmalltalkAgents®

Integrated Development Environment

for

Macintosh

Windows 95/NT



Check out what's new. . .

www.SmalltalkAgents.com

QKS

1405 Main Street
P.O. Box 371478
Montara, CA 94037-1478 USA

© Quasar Knowledge Systems, Inc. 1997
Windows 95 and Windows NT are Trademarks of Microsoft.

Listing 2: Main.java

awake

Set a default value for the visitorName instance variable by implementing the WOComponent's awake method.

```
  public awake()
  {
    visitorName = "David";
  }
```

Listing 3 shows how the Hello page's custom behavior would be implemented in Java. The Hello page is another subclass of WOComponent used as the response to the "sayHello" action defined in the application's Main page.

Listing 3: Hello.java

This subclass of Component defines the custom behavior of the Hello page used as a response for the Main page's action method. It also declares a visitorName instance variable.

```
import next.util.*;
import next.wo.*;

public class Hello extends Component
{
  String visitorName;

  public void setVisitorName (String name)
  {
    visitorName = name;
  }

  public String visitorName ()
  {
    return visitorName;
  }
}
```

Note that using WebScript can reduce the number of lines of code you need to write because it automatically adds default "accessor" methods like those used in the Hello page above to set and get the visitorName instance variable. For example, Listing 4 shows the same Hello page implemented in WebScript.

Listing 4: Hello.wos

This WebScript file has one line — a declaration of the visitorName instance variable. At runtime, this file is parsed by WebObjects and a subclass of WOComponent is dynamically added to the runtime with the developer's custom methods and instance variables.

```
id visitorName;
```

That's it.

When you're done editing your source, you use Project Builder to create the application's executable. **Figure 7** is a screen shot of ProjectBuilder and its build panel being used to create the HelloWorldJava executable.

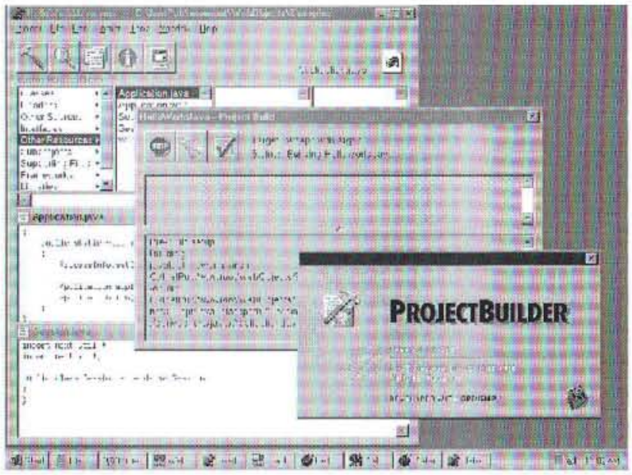


Figure 7. *ProjectBuilder shown here while building the HelloWorldJava application in the background.*

Figure 8 shows how to add database access to the HelloWorldJava application. In **Figure 8**, a Customer entity defined in EOModeler is dragged and dropped into the WebObjects Builder component window creating an instance of a WODisplayGroup that will manage Customer objects fetched from a database. By mapping properties of HTML widgets to attributes and methods provided by the WODisplayGroup, information from the underlying database management systems can be displayed in the user interface — with any business logic being enforced transparently, thanks to the intervening objects.

continued on page 22

Purify on Unix
Bounds Checker on Windows
Now, from the makers of QC, comes...
SPOTLIGHT™
On Macintosh

Find Bugs Fast

Spotlight is the first **Automatic Memory Debugger** for the Macintosh. Instantly detect invalid memory accesses, bad toolbox parameters, leaks, stack overwrites, memory relocation problems, and much more.

Spotlight uses your XSYM file to automatically patch your application — no need to change your source code. No need to recompile. **No learning curve whatsoever.**

The interface gives you instant feedback when an error is detected. You can ignore the error, ignore all future occurrences of the error, or log the error to a text file for later analysis.

Fine Grained Memory Protection

Spotlight identifies reads and writes outside of the application and system heap. But it does not stop there. Spotlight's Object Code Replacement instrumentation **inspects each read and write instruction in your code,** detecting faulty memory reads and writes that occur between blocks, in released blocks, across multiple blocks, and in ROM.

This technology works on any heap object you can allocate: Mac heap objects, C 'malloc' heap objects, even C++ objects created with new.

Spotlight stops your application whenever a faulty memory access is about to occur and displays the **exact source code line** where it will happen. The calling stack is shown along with a display of all variables. A memory viewer shows the erroneous memory address and contents.

Toolbox Validation

Spotlight checks parameters to **over 400 toolbox API calls,** automatically validating handles, memory blocks, return values, and so on. Specific checks catch subtle errors such as drawing into an unlocked GWorld, passing an invalid window pointer, passing an address within an unlocked handle to a routine that may move memory, and too many more to list.

Leak Detection

No more struggling with MacsBug. On program exit Spotlight provides a **clear listing of all leaked memory** showing a full stack trace from where the memory was allocated. Instantly discover all leaked Mac OS objects, malloc'd objects, C++ objects, and resources.

Limitations

Spotlight requires an XSYM file to function. MPW and CodeWarrior users can generate these directly. Symantec users must use ToolServer to link with an XSYM capable linker. Spotlight requires a PowerPC processor.

Spotlight works on the object code. **No source code is required**: just the thing for testing purchased third party libraries.

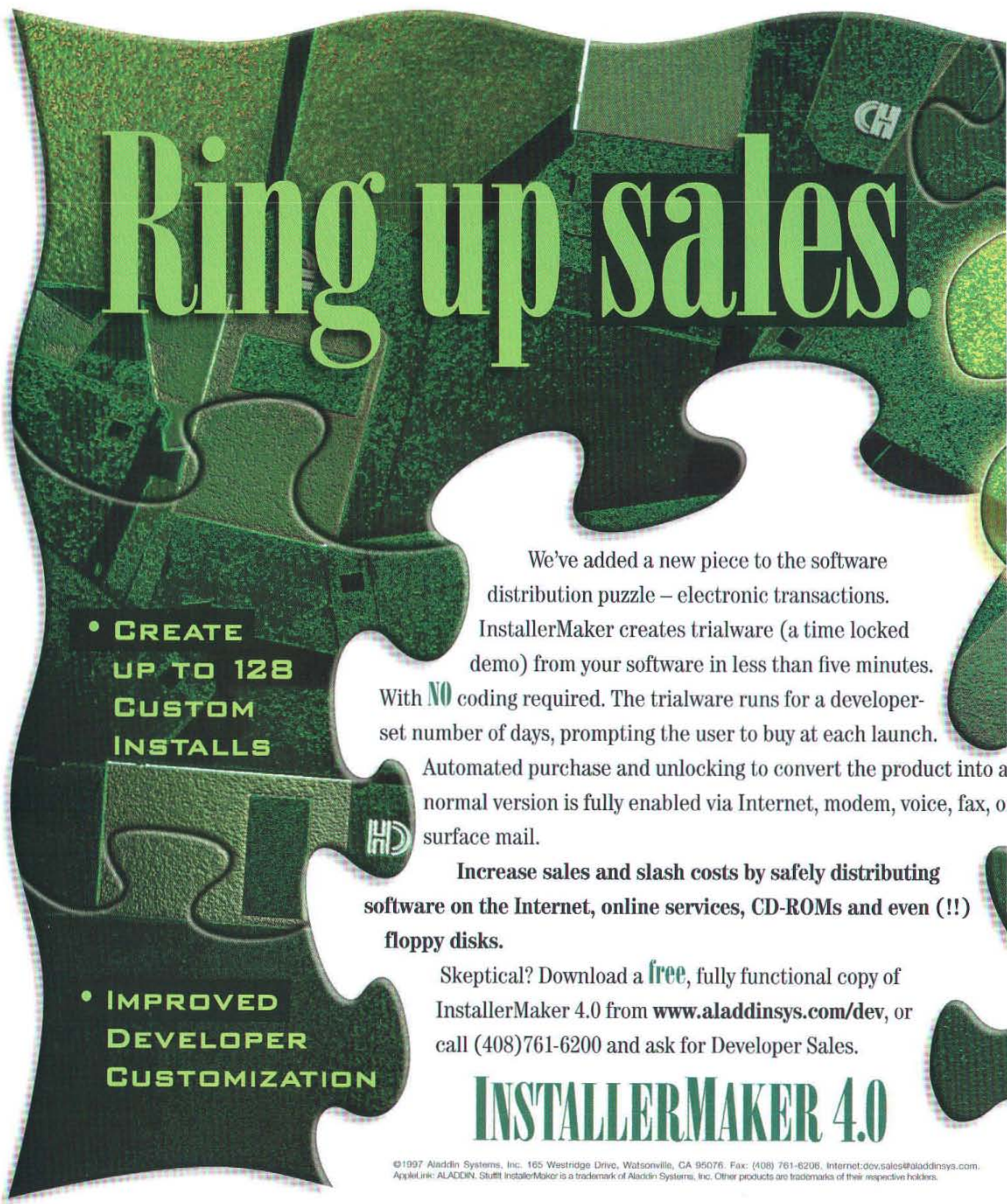
Availability and Pricing

Pricing for Spotlight DR1 is $199 US (plus $5 shipping and handling within the continental US, $15 for international orders). This includes a free upgrade to the GM version and access to ftp interim upgrades leading up to GM. QC users can cross-grade to Spotlight for only $149.

All Onyx products carry a 30 day no questions asked money back guarantee.

Onyx Technology, Inc.
7811 27th Avenue West
Bradenton, Florida 34209

sales@onyx-tech.com www.onyx-tech.com
941 795-7801 941 795-5901 (fax)



Ring up sales.

• CREATE UP TO 128 CUSTOM INSTALLS

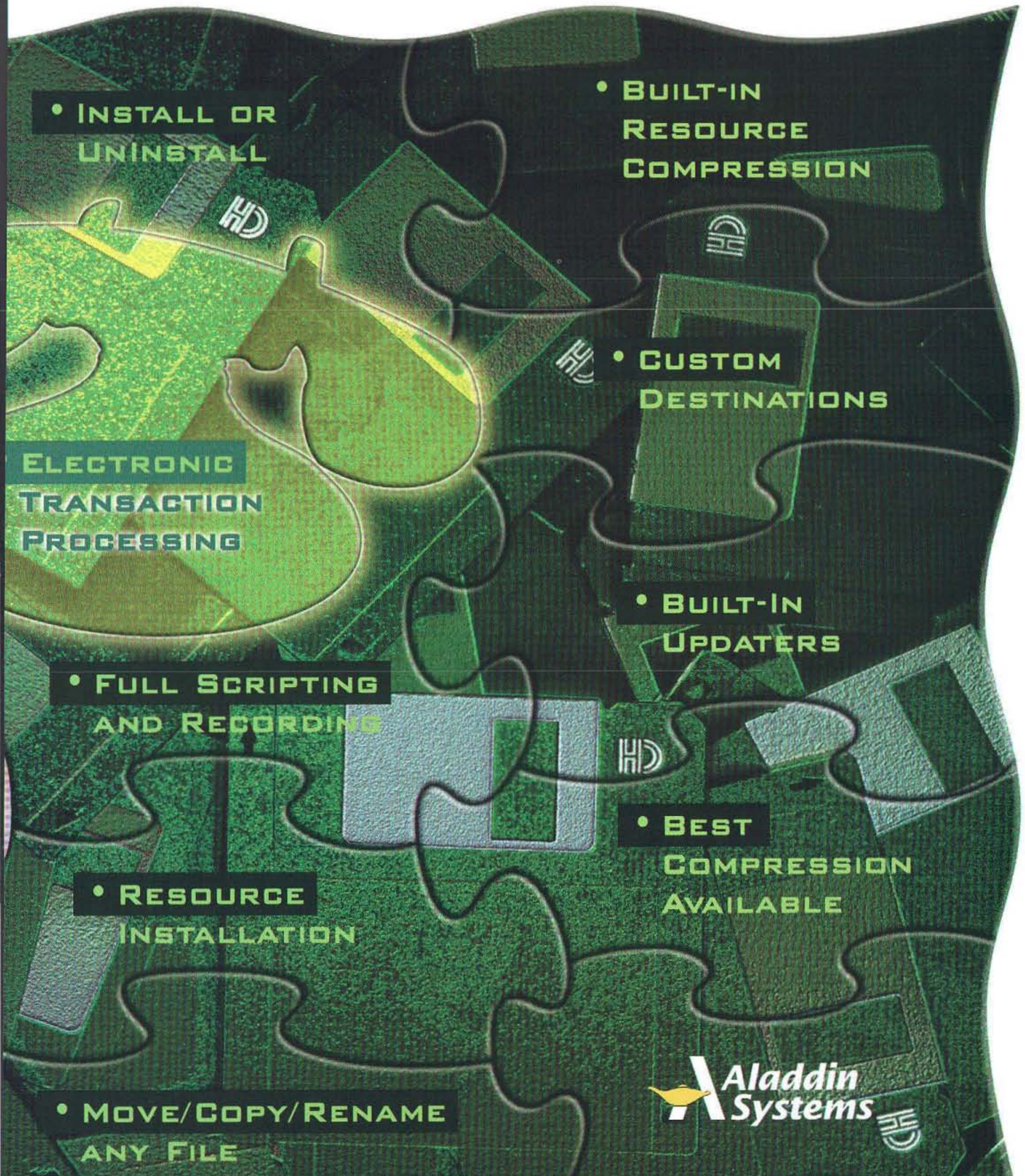
• IMPROVED DEVELOPER CUSTOMIZATION

We've added a new piece to the software distribution puzzle – electronic transactions. InstallerMaker creates trialware (a time locked demo) from your software in less than five minutes. With **NO** coding required. The trialware runs for a developer-set number of days, prompting the user to buy at each launch. Automated purchase and unlocking to convert the product into a normal version is fully enabled via Internet, modem, voice, fax, o surface mail.

Increase sales and slash costs by safely distributing software on the Internet, online services, CD-ROMs and even (!!) floppy disks.

Skeptical? Download a **free**, fully functional copy of InstallerMaker 4.0 from **www.aladdinsys.com/dev**, or call (408)761-6200 and ask for Developer Sales.

INSTALLERMAKER 4.0


• Install or Uninstall
• Built-in Resource Compression
• Custom Destinations
Electronic Transaction Processing
• Built-In Updaters
• Full Scripting and Recording
• Best Compression Available
• Resource Installation
• Move/Copy/Rename any File
Aladdin Systems

continued from page 18

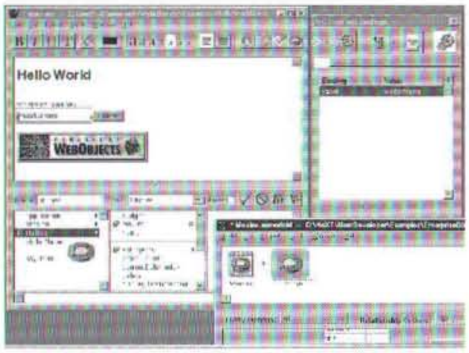


Figure 8. *WebObjects Builder showing how to access objects for entities defined in the EOModeler application.*

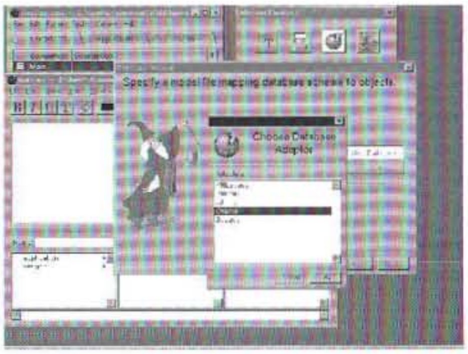


Figure 9. *WebObjects Builder showing abrand new application being constructed with the Database Wizard.*

Figure 9 shows WebObjects Builder while the user is selecting options available in the Database Wizard tool. Specifically, WebObjects Builder is prompting the user for the database type the developer would like to connect to.

HelloWorld is, of course, a very simple application. WebObjects however is most appropriate when building the most complicated applications. **Figure 10** is a screen shot of a non-trivial example application that ships with WebObjects. It's called the WebObjects Monitor, it is used to remotely

WebObjects Class Functional Groupings

In a typical WebObjects Enterprise application there are six workhorse classes from the Foundation Framework that get used all of the time:

```
NSString, NSMutableString
NSArray, NSMutableArray
NSDictionary, NSMutableDictionary
```

However, there are many others you can leverage. Here's a grouping of related classes. Note that the same class has been included under more than one category where appropriate. Also, note that all but the WebObjects Framework is common to OPENSTEP apps. Instead of the WebObjects Framework (optimized Web UI/Web Deployment), OPENSTEP includes the Application Kit Framework (Display PostScript UI, drag & drop, pasteboard services, etc.)

- Operation Processing:
  NSObject, NSInvocation, NSMethodSignature, NSException, NSAssertionHandler, NSRunLoop, NSTimer
- Object Notification
  NSNotification, NSNotificationCenter, NSNotificationQueue
- Object Distribution:
  NSConnection, NSProxy, NSDistantObject, NSRunLoop, NSCoder
- Program Environment:
  NSProcessInfo, NSUserDefaults, NSAutoreleasePool
- Persistance:
  NSCoder, NSArchiver, NSUnarchiver,
- Time:
  NSDate, NSCalendarDate, NSTimeZone, NSTimeZoneDetail
- Thread Control:
  NSThread, NSLock, NSConnection, NSRecursiveLock, NSConditionLock, NSRunLoop
- Structured Data:
  NSArray, NSMutableArray, NSDictionary, NSMutableDictionary, NSSet, NSMutableSet, NSCountedSet
- Data storage as objects:
  NSValue, NSString, NSData, NSMutableData, NSSerializer, NSDeserializer, NSCharacterSet, NSMutableCharacterSet, NSScanner
- Interation
  NSEnumerator

Here are some functional groupings for the WebObjects Framework.

- Request & Response Handling
  WORequest, WOResponse, WOApplication
- State Management
  WOSession, WOSessionStore, WOApplication, WOContext
- HTTP Server Adaptor Management
  WOAdaptor, WODefaultAdaptor, WOApplication
- Page objectification
  WOElement, WOComponent, WODynamicElement,
  Note: there are many subclasses of WOElement provided by the framework that may be used by developers.
- Datasource abstraction
  WODisplayGroup(*)

administer, monitor status, gauge performance, and provide fail-over for your application's instances during the deployment phase of your project.

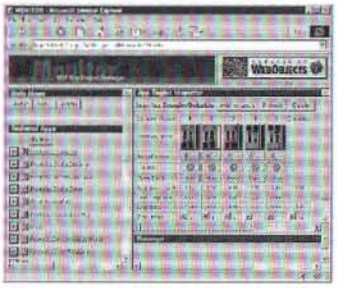


Figure 10. *WebObjects Monitor. An example of non-trivial WebObjects application. This particular app is used to administer a WebObjects application during deployment. In the above screen shot, we see that the DodgeLite application configuration is being inspected. The detail information shows that it has currently been configured with four instances (two of which are running).*

Summary

WebObjects provides frameworks to give your app a logical head start, deployment infrastructure to give your application a scalable structure, and development tools to help you create, maintain, and extend your application throughout its life-cycle. Today the tools run on NeXT's OPENSTEP/Mach operating system as well as Microsoft's Windows NT. They will also be available soon on the next-generation Mac OS, Rhapsody, on top of PowerPC hardware. Regardless of what platform you use to develop your application, you can enjoy portability when you decide to deploy it. The WebObjects deployment runtime is available on Windows NT, OPENSTEP/Mach, Solaris, and HP-UX.

For detailed information about using WebObjects, class documentation, user guides, and training schedules, visit NeXT's web site at <http://www.next.com>.

In a typical development scenario, virtually none of the Enterprise Objects classes described below in the Access and Control layers will appear in your actual source code (but the point is they can if need be for customization). The exceptions are the EODataSource and EOQualifier classes. The modeling classes are configured using the EOModeler application, the Adaptor classes are encapsulated in your RDBMS adaptor, much of the other classes are created by other classes for you and just work behind the scenes. Their APIs are available if you need them, and they may be subclassed, categorized, and/or hooked up to delegates to fine tune their behavior.

In the Enterprise Objects Access Framework there are these class functional groupings:

- Accessing specific relational datastores
  EOAdaptor, EOAdaptorContext, EOAdaptorChannel, EOSQLQualifier, EOAdaptorOperation
  Note, subclasses of these classes are provided by NeXT for Oracle, Sybase, Informix, and ODBC client libraries. In addition, an example with complete source is provided for a version that uses the file system (FlatFileAdaptor). Source is also provided to the ODBC adaptor so developers can write their own custom adaptors if they like using this adaptor as an example basis.

- Mapping any custom adaptor to objects
  EODatabase, EODatabaseContext, EODatabaseChannel, EODatabaseOperation
- Modeling an object graph to a RDBMS schema
  EOModel, EOModelGroup, EORelationship, EOAttribute, EOEntity, EOJoin, EOEntityClassDescription
- Accessing Database Metadata
  EOSQLExpression, EOAdaptor, EOAdaptorChannel
- Store procedure utilization
  EOStoredProcedure

In the Enterprise Objects Control Framework, there are these class functional groupings.

- Object Graph Change and Transaction Management
  EOEditingContext, EOUndoManager, EOObserver, EOObserverCenter, EOObserverProxy, EODelayed-Observer, EODelayedObserverQueue
- Object Storages Abstraction
  EOObjectStore, EOGlobalID, EOFault, EOFaultHandler, EOObjectStoreCoordinator, EOCooperatingObjectStore
- Object uniqueing ID stamping
  EOTemporaryGlobalID, EOGlobalID
- Object Querying
  EOKeySortOrdering, EOFetchSpecification, EOQualifier, EOSQLQualifier, EOAndQualifier, EOOrQualifier, EONotQualifier, EOKeyValueQualifier
- Simple Source of Objects for WODisplayGroups
  EODataSource
- Enterprise Objects
  Any class you like with any inheritance scheme you like.

by Dave Klingler

What's Inside OPENSTEP... Really?

An Introduction to OPENSTEP's Development Tools

The next several pages will be an attempt to give you a complete overview of OPENSTEP's development system as it might pertain to Rhapsody. Of course, we can't quite cover everything, but hopefully when you're done reading you'll know a little bit about OPENSTEP's basic parts and why developing under OPENSTEP has brought a happy grin to the faces of so many lucky programmers. OPENSTEP isn't perfect, but it's a well-designed springboard for a Macintosh programming renaissance during the next few years.

The major elements of OPENSTEP are the Foundation Kit, Interface Builder, Display Postscript, and Objective-C. There are some important minor players as well: ProjectBuilder, Header Viewer, Librarian and pswrap. All of these elements are probably unfamiliar to you unless you've used NEXTSTEP or OPENSTEP before, so let's start by taking a quick run through the development process under OPENSTEP to see where the pieces fit. We'll then go back to take a closer look at the major players.

The Development Process

The development cycle under OPENSTEP is fast enough that it tends to leave you wondering what you skipped when you're finished. I'll outline my own development cycle here; you'll probably evolve your own techniques once you start programming under OPENSTEP.

I usually begin by designing classes. This process is similar in any modern language; I figure out what data my program will use and derive a rough list of the major classes involved. Optimizing for speed versus efficiency, or server storage versus local storage, maintainability or even communications bandwidth will influence the design of my class system.

Over a period of about my first two years working with Objective-C I noticed that my techniques for designing new objects changed radically and often. I attribute that to the seemingly subtle, but huge differences between designing for Objective-C and for conventional languages like C or Pascal or even quasi-oop languages like C++. I take heavy advantage of dynamic typing and dynamic binding, which are not available in most languages other than Smalltalk. Java is currently mostly statically typed as well, but JDK 1.1 has reflection, a clear sign that the language is evolving in the right direction.

Bizarrely enough, I've gotten to the point where I write almost all my overview documentation before I write most of the code. I'll write test code to make sure a concept's going to work if it's something radical, but for the most part the code's completely described before it's written. I'm finished when any competent Objective-C programmer could write my app from my docs.

The documentation process is made easier by the fact that every Objective-C class is comprised of an interface file and a class implementation file. The interface file is a fancy header file that contains all the class variables that might be needed by a programmer using the class, plus prototypes for class methods and functions. When I'm designing a new class, of necessity I have to decide how that class will interact with the outside world. Whatever's externally available goes in the interface file and, with a comparatively tiny amount of work, an explanation goes with it into the documentation. Now I've got my interface files and all the documentation someone would need to use any of the classes in my new app.

At this point I'll either hand it off to a group of programmers for them to complete or paste the method and function prototypes

Dave Klingler is a computer systems architect who lives in Albuquerque, New Mexico with his two dogs, Bruce and Shasta. He is currently working on virtual reality tools under OpenStep.

from the interface into a new file called the class implementation file and begin filling out the class. This is the file that will actually contain the code for the class, and if you want to sell your classes you can compile this file and sell the result along with the interface file and the documentation. No one gets your source code, but they do get an explanation of all the classes' externally available variables and methods. Their classes can interface with your classes without knowing what's inside them, and that, after all, is what object-oriented programming is all about.

Lots of people in the OPENSTEP world sell classes instead of applications. One of the first classes I bought was BenaTong's Serial class, so that I could save some work writing a telecommunications package.

Because I've got a list of method prototypes to implement, I can just work my way through the file coding each one. When I reach the last one the class will be finished and quite often it "just works" with little or no debugging.

After designing the classes, it's time to pull out ProjectBuilder. ProjectBuilder manages the process of building a new application, module or palette under OPENSTEP. I tell ProjectBuilder to create a new project, create a few icons for the program and data files and drop them into ProjectBuilder's various boxes, drop in the new classes and double-click the "nib" file that ProjectBuilder has created for me. "Nib" stands for NeXT InterfaceBuilder, and that file contains the elements of the interface for the program, arguably most of it. Using Interface Builder might be compared to using a graphical version of ResEdit, but InterfaceBuilder does much more.

Next I design the rest of the program. Keep in mind that most of the elements in the average program other than the ones you've just designed (and they're usually very few) are already running in the OPENSTEP OS. This means that you can use "live" versions of those objects when you design your program's user interface by just grabbing what you want off a palette. You can also put your new classes into their own palette and use them too.

InterfaceBuilder allows you to design a program and "run" it without actually ever compiling it, because the code for any objects you're using other than your own has already been compiled. That code is running or at least available in the operating system as a shared library. When you've finished "drawing" your program the way you want it to work, including the classes you can't see but must be integrated anyway, you select "test interface" from InterfaceBuilder's menu and InterfaceBuilder runs the program for you without classes that it doesn't have yet. It gives you a big Frankenstein-style switch that you can use to shut the program down if your experiment doesn't like to quit. If you don't like the user interface, you can play with it until it's easy to use or does exactly what you wanted. It's far more powerful than an ordinary interface design program because you effectively have a graphical window into the guts of your app.

To release my application in other countries I can design nibs in other languages. OPENSTEP has support for English, French, Spanish, Swedish, German and Japanese. Most of the time it's just a matter of changing the words on the menus. There

ObjectSet Mail SDK

Mail enabling made easy

It's never been this easy to add custom messaging to your applications. Send and receive mail. Handle attachments. Just use ObjectSet.

Ideal for Internet/Intranet environments, ObjectSet gives Macintosh, Be and Windows developers reliable APIs for MIME, SMTP and POP3 protocols. ObjectSet's royalty-free software components are perfect for developing applications which require integrated messaging services, from commercial software packages to custom corporate applications. With ObjectSet, you benefit from mail protocols without having to learn, code and debug them on your own.

Come take a closer look at our libraries, documentation and demo applications. They're on our Web just for you (http://www.smartcodesoft.com). Download a free copy today and see why ObjectSet APIs are the easiest you'll ever use.

ObjectSet Mail SDK key features

- MIME, SMTP and POP3 APIs for Macintosh, Be and Windows
- C++ class libraries
- Framework integration using PowerPlant, MacApp, MFC
- Documentation, samples and reusable code
- MIME encoding/decoding: Base64, Quoted-Printable, UUDECODE, BinHex, AppleSingle, AppleDouble
- Send messages using SMTP, retrieve messages using POP3
- Supports OpenTransport, MacTCP, BSD Sockets, WinSock DLL
- Subscription price includes next two upgrades free
- Portable multiplatform version and source code also available

SMARTCODE SOFTWARE

Smartcode Software, Inc.
Phone: (847) 945-3516
Fax: (847) 945-6889
http://www.smartcodesoft.com
mtsales@smartcodesoft.com

In Europe:
Phone: +33 4 67 59 30 40
Fax: +33 4 67 59 30 48
http://www.smartcode.fr
mtsales@smartcode.fr

are localization firms that specialize in taking an English nib file and using it to create other language-specific versions of the program. They'll send back the nib and you can drop it into ProjectBuilder. They'll also translate other messages that reside in your code if it's needed. In most cases your app will "just work" in any language your customer chooses to use.

When I finish with InterfaceBuilder (and I might reverse the process and play with InterfaceBuilder first because it's sometimes more productive) I tell ProjectBuilder to compile the app. I'll select "debug" first if the app is large or I'm trying something fancy. If I'm debugging, ProjectBuilder will build the app with all its nibs and then compile the new classes with debugging extensions. It'll then drop me into gdb (a debugger) with the app loaded, where it's up to me to debug. There are various other debugging tools available under OPENSTEP that allow me to examine the app's messaging, memory usage, optimization, etc. Quite often if I'm really bamboozled I'll tell gdb to show me all the messaging taking place between objects. It's something like watching your children begin to talk.

I haven't yet discussed Postscript wraps or the cool process of trying out your Postscript code in Yap.app (there's a server running in the os, so why not?) before it goes into your project. ProjectBuilder knows about wraps and various other resources too, so it's an extremely useful tool for managing the process of creating a large application. Let's take a look at wraps and Display Postscript in general.

Display Postscript

Display Postscript (DPS) did not exist when NeXT first got together with Adobe to design a common language for writing to the screen and the printer. The general idea was to achieve true WYSIWYG by using the same code to describe both, and so DPS was born. Steve Jobs waxed poetic about XWindows while describing why NeXT had chosen DPS over X11; he called X "brain damaged". It is true that like many groundbreaking efforts, X became a little outmoded, and DPS is an elegant system. DPS is, however, imperfect in the context of the new age of multimedia, so we'll probably see many additions to DPS over the next few years.

Depending on your program's performance needs, there are various methods available for adding Display Postscript to your programs. Note that most programs use very little DPS, and unless you're designing a drawing application you probably won't have to learn much. That said, playing with Postscript interactively can be somewhat addictive, like fooling with turtle graphics and Logo if you're old enough to remember. You may choose to spend more time with it than is absolutely necessary, and if you do, the principles are the same.

You may or may not know that Postscript is a stack-based client-server page description system. Your program is the client, and the server is effectively either the screen or the printer. In theory it would be nice to write code that worked on both screen and printer; in practice you can't rely on all printers being Level II Postscript-compatible, so you may have to rewrite small sections of your display code for safe printing if you want to use Level II features.

There are three basic methods for using Display Postscript: operator functions, wraps and user paths, in descending order of their execution times. Operator functions are simplest, and you'll use them when you want to make a quick call to the server for a single function. Here's an example of a common combination of commands in Postscript, a "moveto" command followed by a "lineto" command. They create what is known in Postscript as a "path, and they're followed by an instruction to "stroke" the path, or fill it with ink.

```
10 10 moveto
50 50 lineto
stroke
```

As mentioned, one of the really fun parts of OPENSTEP is getting into Yap (Yet Another Postscript processor) and playing with Postscript interactively. Some time ago I ported BattleZone to NEXTSTEP, and I threw the entire opening screen into Yap to test it. I then moved elements of it around until I was happy with the way the screen looked. It was so much fun it almost made up for the rest of the porting process.

You can include these instructions in your code by using PS operator functions. OPENSTEP has two PS library functions for every PS operator, one for drawing in a default context and one that allows you to specify the context. Here's the equivalent of what's above, using the PS operator functions in psops.h, which draw in a default context:

```
PSmoveto(10, 10);
PSlineto(50,50);
PSstroke;
```

Now you've seen the original Postscript and the way you can implement it using single operator functions. As you can see, it is easy to convert any Postscript calls into operator function calls and insert them into your program. The problem with doing that is that each one of these calls is a separate message to the server. If you're calling one function occasionally, you're sending one message to the server occasionally, which is as good as you're going to get. For more than one line of Postscript, however, you'd do better to package your Postscript into a "wrap".

A wrap packages your calls into a C function library that ProjectBuilder can be told to include at compile time. You'd write a text file with the following in it:

```
definepsPSWDefs()
/ML {% X1 Y1 X Y
   moveto
   lineto
} bind def
endps

defineps multiline(float data[x]; int x; int length)
  data
  1 1 length ML for
endps
```

Drop this into ProjectBuilder, which will call a program called "pswrap", to turn the file into a friendly, eminently readable and efficient package that looks like this:

```
/* ./sym/c_gpr.c generated from c_gpr.psw
 by unix pswrap V1.009 Wed Apr 19 17:50:24 PDT 1989 */
#include <dpsclient/dpsfriends.h>
#include <string.h>
#line 1 "c_gpr.psw"
#line 10 "./sym/c_gpr.c"
void PSWDefs( void )
{
 typedef struct {
  unsigned char tokenType;
  unsigned char topLevelCount;
  unsigned short nBytes;
  DPSBinObjGeneric obj0;
  DPSBinObjGeneric obj1;
  DPSBinObjGeneric obj2;
  DPSBinObjGeneric obj3;
  DPSBinObjGeneric obj4;
  DPSBinObjGeneric obj5;
  char obj6[2];
  } _dpsQ;
 static const _dpsQ _dpsF = {
  DPS_DEF_TOKENTYPE, 4, 54,
  {DPS_LITERAL|DPS_NAME, 0, 2, 48},/* ML */
  {DPS_EXEC|DPS_ARRAY, 0, 2, 32},
  {DPS_EXEC|DPS_NAME, 0, DPSSYSNAME, 14}, /* bind */
  {DPS_EXEC|DPS_NAME, 0, DPSSYSNAME, 51}, /* def */
  {DPS_EXEC|DPS_NAME, 0, DPSSYSNAME,107}, /* moveto */
  {DPS_EXEC|DPS_NAME, 0, DPSSYSNAME, 99}, /* lineto */
  {'M','L'},
  }; /* _dpsQ */
 register DPSContext _dpsCurCtxt = DPSPrivCurrentContext();
 char pad[3];
 DPSBinObjSeqWrite(_dpsCurCtxt,(char *) &_dpsF,54);
 if (0) *pad = 0;   /* quiets compiler warnings */
}
#line 7 "c_gpr.psw"
#line 43 "./sym/c_gpr.c"
void multiline(const float data[], int x, int length)
{
 typedef struct {
```

A.D. Software

You think in C++

So does OOFILE

Cross-Platform
Mac, Windows, OS/2, Unix

Database
Object-oriented, c++ syntax
Faircom c-tree Plus®
dBase III+ and IV

Report-Writer
Mac & Windows WYSIWYG
RTF & HTML

Graphing
Standalone or within reports

Frameworks
PowerPlant & MFC

Code-Generation
AppMaker and OOFILE create complete database applications

dent@highway1.com.au
www.highway1.com.au/adsoftware/

also available at DevDepot

Serious development tools

for... • Object Oriented programmers
• Cross-platform developers
• Help and Multimedia authors

Mac2Win™

The solution for porting Mac applications to Windows **saves...**

Time	Get to Windows in 1/3 to 1/2 the time
Money	Leverage your existing investment in Mac source code
Code	Support for PowerPlant, TCL, MacApp and System 7
Heads	One set of source means no maintenance headaches

Object Master™
Object Oriented programming environment

Mac2Win™
Port Mac apps to Windows

QuickHelp™ 4.0
WinHelp for the Mac

QuickView™
Multimedia Viewer for the Mac

Altura Software, Inc
510 Lighthouse Ave. Suite 5
Pacific Grove, CA 93950

Voice: (408)655-8005
Fax: (408)655-9663
Web: http://www.altura.com

```
 unsigned char tokenType;
 unsigned char topLevelCount;
 unsigned short nBytes;
 DPSBinObjGeneric obj0;
 DPSBinObjGeneric obj1;
 DPSBinObjGeneric obj2;
 DPSBinObjGeneric obj3;
 DPSBinObjGeneric obj4;
 DPSBinObjGeneric obj5;
 } _dpsQ;
typedef struct {
 char obj6[2];
 } _dpsQ1;
static const _dpsQ _dpsStat = {
 DPS_DEF_TOKENTYPE, 6, 54,
 {DPS_LITERAL|DPS_ARRAY, 0, 0, 48}, /* param[var]: data */
 {DPS_LITERAL|DPS_INT, 0, 0, 1},
 {DPS_LITERAL|DPS_INT, 0, 0, 1},
 {DPS_LITERAL|DPS_INT, 0, 0, 0},   /* param: length */
 {DPS_EXEC|DPS_NAME, 0, 2, 48}, /* ML */
 {DPS_EXEC|DPS_NAME, 0, DPSSYSNAME, 72}, /* for */
 }; /* _dpsQ */
static const _dpsQ1 _dpsF1 = {
 {'M','L'},
 }; /* _dpsQ1 */
_dpsQ _dpsF; /* local copy */
register DPSContext _dpsCurCtxt = DPSPrivCurrentContext();
char pad[3];
register DPSBinObjRec *_dpsP = (DPSBinObjRec *)&_dpsF.obj0;
register int _dps_offset = 48;
_dpsF = _dpsStat;  /* assign automatic variable */
_dpsP[0].length = x;
_dpsP[3].val.integerVal = length;
_dpsP[0].val.arrayVal = _dps_offset;
_dps_offset += x * sizeof(DPSBinObjGeneric);
_dpsP[4].val.stringVal = _dps_offset;
_dps_offset += 2;
_dpsF.nBytes = _dps_offset+4;
DPSBinObjSeqWrite(_dpsCurCtxt,(char *) &_dpsF,52);
 DPSWriteTypedObjectArray(_dpsCurCtxt,
    dps_tFloat, (char *)data, x);
 DPSWriteStringChars(_dpsCurCtxt,(char *) &_dpsF1,2);
 if (0) *pad = 0;   /* quiets compiler warnings */
}
#line 12 "c_gpr.psw"
```

As you can see, it's easier to work in Postscript and let ProjectBuilder or pswrap do the work for you. It's also not a good idea to look too often at the files that pswrap generates, because you'll begin losing sleep.

There's a third way to interact with the DPS server, and that's with user paths. A user path is a way of packaging up a bunch of Postscript and translating it beforehand for the server. You can go even further by telling the server to store it in a user dictionary. That way you can make one call to the server, perhaps to supply arguments, and you minimize your program's messaging overhead. The technique is available when you need it, but from experience I can tell you that it's easier to use wraps and then go back and optimize where necessary with user paths. User paths are sometimes not very easy to debug.

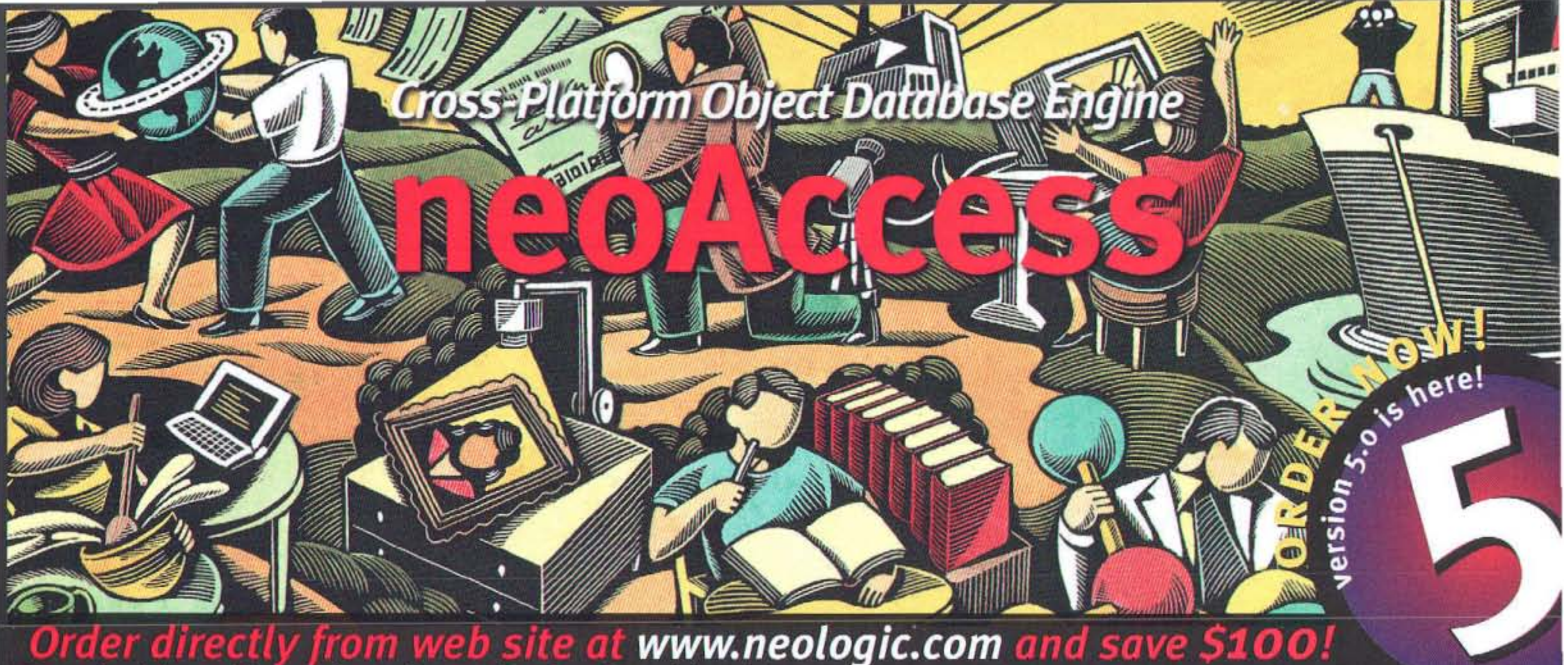
For the most part whenever you use Postscript, you'll be drawing in a "View" object, a part of the Application Kit that maintains its own state for Postscript operations. Let's go back for a look at InterfaceBuilder and the Appkit and see why InterfaceBuilder plays such a large part in OPENSTEP programming.

INTERFACE BUILDER

On the surface InterfaceBuilder looks essentially just like any interface builder. What it's doing underneath is far more rich and, in the end, far more useful to the programmer. InterfaceBuilder allows you to assemble all of the elements of your program and test them, in some cases without even compiling the program. You do this by grabbing objects off a palette and placing them where you want them. You can use almost any object in the Application Kit, which is an extremely rich source of material, or you can also make up palettes of your own objects and drop them into InterfaceBuilder.

A side benefit of using InterfaceBuilder is that much of your program will automatically conform to the OPENSTEP user interface standards. InterfaceBuilder doesn't limit you to creating applications. You can create modules as well, for use in other apps. After you're done bringing in the various elements of your program, InterfaceBuilder allows you to define the relationships these elements will use to work together. When you're finished, you can test your application without compiling it from within InterfaceBuilder. If a program works in InterfaceBuilder, it usually works after it's been compiled. Sometimes you'll even find yourself writing programs that never even run outside of InterfaceBuilder.

There are three major parts of InterfaceBuilder other than the area in the middle of the screen where you assemble your apps. The first is the file window (**Figure 1**), where all the different parts of your app are kept, including the parts that have no visual interface. As you add pieces to your application you'll see icons for each instance appear in the file window. You can also instantiate your own objects as you write code for them and they will appear here.



Order directly from web site at www.neologic.com and save $100!

Download the architectural overview or view it on-line with Adobe Acrobat 3.0

The most powerful object-oriented database engine available

NeoAccess® displays electrifying performance—up to ten times that of the competition. Behind its elegant programming interface is a fully optimized relational query engine built for speed. It also has an object cache with automatic garbage collection so your applications can run in a much smaller memory footprint than you ever imagined possible. And NeoAccess has no runtime fees so you pay one affordable price no matter how many copies of your application you use or sell.

While others promise cross-platform – NeoAccess delivers. NeoAccess is a set of C++ classes designed for use with popular compilers and frameworks on Windows®, Macintosh®, and Unix™ platforms. Thousands of developers, including America Online® and Claris®, have already found that NeoAccess enabled them to build fast, powerful internet applications in record time. That's why there are more NeoAccess based applications on end-user machines than any other object database backend.

M–F 9AM to 5PM Pacific:
1-800-919-6353
For Information and Customer Service:
1-510-524-5897

neo·logic®

NeoLogic Systems, Inc.
1450 Fourth St., Suite 12, Berkeley, CA 94710
v. 510.524.5897 f. 510.524.4501
neologic@neologic.com

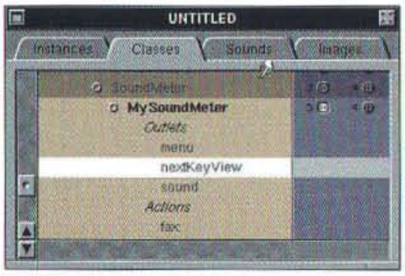
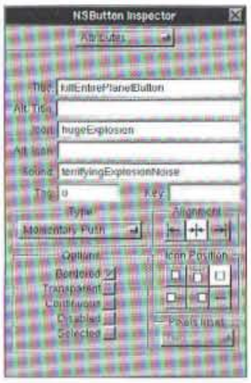


Figure 1. The File Window contains all the pieces of an application.

The file window keeps track of sounds, images, and information about the different classes available to you as well. You can use the file window's class browser to browse through the classes for a particular class. The class browser will allow you to parse new classes into InterfaceBuilder for use with the other objects. You can also subclass existing classes in the file window, and InterfaceBuilder will write skeleton code for your new subclass, including any action methods or outlets (pointers to other objects) your subclass might have. You can add these action methods and outlets in the Inspector window. Inspector windows are one of the most important user interface innovations in OPENSTEP, and a number of MacOS and Windows programs have begun using them. Essentially an inspector gathers all the important information about an object in one place. InterfaceBuilder makes fundamental use of inspectors in a most artful manner.



Title: killEntirePlanetButton
Alt. Title:
Icon: hugeExplosion
Alt. Icon:
Sound: terrifyingExplosionNoise
Tag: 0
Key:
Momentary Push
Bordered [✓]
Transparent
Continuous
Disabled
Selected

Figure 2. An InterfaceBuilder Attributes inspector window.

Figure 2 shows an InterfaceBuilder Attributes inspector for a fictitious button in a fictitious application. It's what you'd see if you selected the button in the same manner as selecting a line in a drawing program. The Attributes inspector allows you to change the title of the button instance, the icon it displays, the sound it makes, what kind of button it is (momentary push, pushOn/pushOff, etc.), and various other attributes. You can

Late Night SOFTWARE

ScriptDebugger

Completely Replaces Apple's Script Editor

Script Debugger is a powerful and flexible AppleScript™ authoring tool that lets novice and seasoned script writers get the most from AppleScript. By combining an easy-to-use interface with extensive development tools, it makes script writing faster and easier than ever before.

Script Debugger, recently awarded a five-mouse rating in *MacUser* (UK edition), offers:

- **No 32K limit on scripts**
- **True single-step execution of all AppleScript scripts**
- **A powerful scripting enviroment that includes Drag & Drop editing**
- **Power PC Native**
- **Scriptable and attachable**

◆◆◆◆

"...the program [Script Debugger] is a solid performer, and its support for large file sizes, helpful Dictionary window, superb scripting additions and complete scriptability make it a very good choice."
Avi Rappoport and Ed Allen
***MacWEEK* Magazine**

Keep current with your Script Debugger upgrades — visit our Web site: http://www.latenightsw.com

$129

Late Night Software Ltd.
Voice (604) 929-5578
Fax (604) 929-4961
E-mail gtubin@wimsey.com

either specify sounds or icons by typing their names into the inspector or you can drop them on to the button and they'll appear here. InterfaceBuilder's Inspector Window carries all the important information about an individual object.

In addition to attributes you'll see other common elements such as details on the object's connections to other objects (messages it might receive or send, and to or from whom), the object's size in various dimensions, or help attachments to an interface element. In short, the Inspector Window contains detail. Last but not least is the Palette Window. It's the most intuitive window to use, because you simply drag objects from the palette to the application window or panel in which they should reside. You can drag Window and Panel objects, too.

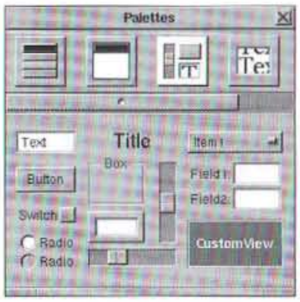


Figure 3. *InterfaceBuilder's Pallette window.*

When you've finished dragging, browsing, parsing, clicking and typing, it's time to finish your application by drawing connections between all the different objects. When you've finished, your objects will be able to send messages to one another, and you can test them by selecting "Test Interface" in the menu. When you're all finished, click "Save" and InterfaceBuilder will archive the objects for you in a nib file. The nib contains most of the information about a program, and as you can see, anything in the nib is machine non-specific. That's why apps written in OPENSTEP are easily delivered across processor architectures. InterfaceBuilder is very cool.

OBJECTIVE-C, ALIAS OBJECTIVE-C++

Fear not Objective-C. Yes, it is another programming language. It will, however, give you a competitive advantage in development that you probably don't believe possible right now. It's also easy to learn, and before you mumble to yourself that you've heard that before, let's take a look.

Objective-C was designed by Brad Cox, Smalltalk guru and the author of "Object-Oriented Programming, An Evolutionary Approach", a seminal work in OOP circles. He chose to make Objective-C a mix between Smalltalk and C, attempting to keep the best features of both. Objective-C takes dynamic typing and dynamic binding from Smalltalk. Within a class, however, it preserves the conciseness and efficiency of C.

On the surface, there are basically three differences between Objective-C and C:

1. Objective-C has a type called "id" that functions as a pointer to a generic object.
2. id is the default return type, rather than an int.
3. Objective-C has a messaging syntax for calling a method within an object. It uses embedded arguments, and it looks like

```
[object method]
```

Here's an example of a message to a hypothetical object called **graphObject** to plot a point at location (x,y,z) with color **graphColor**:

```
[graphObject plotXAt:x andYAt:y andZAt:z
  withColor:graphColor];
```

The syntax may seem strange with the embedded arguments thrown in, but it's not all that confusing. The method could be prototyped as

```
- (void) plotXAt: (float) x
     andYAt: (float) y
      andZAt: (float) z
     withColor: (float) color;
```

The idea is that the format is more conducive to designing readable code. You can give your methods sentence-like names like the one above. I've been able to return to code I haven't seen for over a year and begin where I left off. I have never found another programming language that allows an ease of overall maintainability that compares to that of Objective-C. Syntax, however, may make Objective-C maintainable, but that isn't what makes it powerful. What makes it powerful is its features, and my favorite features of Objective-C are dynamic typing and binding.

The utility of a generic pointer to an object is that it gives the programmer the ability to point to an object without knowing what it is beforehand. Objective-C has reflection and discovery, so I can point to an unknown object and ask it to describe itself and whether it knows how to do certain things.

By using id to point to the object rather than statically typing it at runtime as an object of class X (you can do that too), I'm telling my program not to worry about what the object is yet. The object will be bound at runtime, and permiting me to choose what object I want during the program's execution instead of being forced to predict it at the link stage. The object I bind with at runtime could be written years after the calling code, and the calling program wouldn't know the difference.

Dynamic binding, dynamic typing and object discovery and reflection allow me to utilize true polymorphism. I might have several different code libraries containing classes that all have the same names but do different things. I might have a method call in a class, for instance, that looks like

```
[storageObject insertRecord:record];
```

One application might call for the storage method to be a hashtable. Another application might be best suited with a stack. The same class could be used for both, without any rewriting, because my stack class and hashtable class understand the same methods. All of this flexibility might be hard to manage, except that Objective-C allows me to declare formal protocols for classes to obey. A class declaration in Objective-C looks like

```
@interface BlowUpTheWorldClass : Object
```

The same class declaration with an instruction to the compiler to make sure that the class follows a formal protocol looks like

```
@interface BlowUpTheWorldClass : Object < BlowsUpThings,
CannotBeStopped >
```

Protocols are truly powerful because they allow the programmer to easily take advantage of extreme polymorphism. A class can send the same message to classes of entirely different inheritance subtrees, which leads us to inheritance and scope coherency.

Scope coherency is not an issue with Objective-C; it uses a single inheritance model. If it's necessary to inherit methods of a wildly disparate nature that don't fit into a single inheritance model easily (and my experience has been that this case is rare), I can group methods into Categories. Categories aren't only good for that purpose, however; they're an extremely powerful tool with many uses.

Using a category, I can add methods to a class without having its source code. Effectively, I can extend classes that are already running, and all subclasses of those classes will inherit the abilities you added transparently.

I can also group like methods in a very large class into various categories. That ability nets me incremental compilation, better locality of reference, configurability of classes for special needs, and a way to divide up a large class among multiple developers.

HeapManager™

Biit, Inc.

Fast, affordable memory allocation library and debugging tool for Macintosh applications written in Metrowerks C/C++

- 12-100 times faster than compiler allocators
- debugging api for finding memory errors
- external debugging tool to track memory usage, display errors and control the allocator while your application runs

$99

See our website at http://www.biit.com for more information and ordering instructions

With OPENSTEP, NeXT brought many of the best features of C++ into Objective-C, renaming the language Objective-C++. The end result is an extremely powerful marriage, sort of like C++ with a clean syntax and much-improved maintainability.

Some members of the Java team originally worked for NeXT, and Java resembles Objective-C far more closely on a structural level than it does C++. If you've looked into Java, and liked it, you'll love Objective-C.

FOUNDATION KIT

Development under NEXTSTEP has always basically been AppKit + Objective-C + DPS, which somehow magically added up to more than the sum of their parts. Over the years, though, there were many NEXTSTEP developers who argued that NEXTSTEP was somehow impure in a few ways. It lacked garbage collection. It had no string objects. It didn't support Unicode. Many of the problems we discussed were along the lines of the perennial "why should the user throw the disk away to eject it?" discussion in the Mac world.

With OPENSTEP, NeXT had a chance to redo some things in a better fashion. Unfortunately, they were still under time pressure, which meant that some of the changes weren't optimal solutions. Still, OPENSTEP makes some good improvements on an already elegant foundation. NEXTSTEP was the first step; OPENSTEP is the next step.

Foundation Kit brings some final polish to the OPENSTEP development environment. It brings some basic utility classes, like NSArray, NSThread, and NSException. It brings reference counting. It brings Unicode strings. It brings... complexity. It brings slowness and bulk. Such is the price of power and maturity.

Mac developers are already familiar with Unicode. Foundation Kit defines the following encodings supported by NSString classes:

Encoding:	Purpose:
NSASCIIStringEncoding	for strict 7-bit ASCII encoding within 8-bit chars
NSEUCStringEncoding	for Japanese text
NSISOStringEncoding	for ISO Latin 1
NSNEXTSTEPStringEncoding	for 8-bit ASCII encoding with some extensions
NSNonLossyASCIIStringEncoding	(undefined)
NSSymbolStringEncoding	(undefined)
NSUnicodeStringEncoding	standard Unicode encoding for string objects
NSUTFStringEncoding	8-bit Unicode for transmission by ASCII-based systems

If you produce software that must be localized, you'll have plenty of fun with these. There's an encoding for every purpose. Support for Unicode is important, but what's arguably an even bigger addition with Foundation Kit is that it brings garbage collection and standardized exception handling.

NeXT's solution to garbage collection is reference counting — possibly not the most elegant solution they could have chosen, but it's robust and it works. Basically the Application object for each application keeps an autorelease pool for storing references to objects. When an object is allocated or copied from another object, it is given a reference count of 1. Every message from another object to retain an object increments the reference count; every message to "autorelease" the object decrements it. When the reference count reaches 0, the autorelease pool releases the object.

To understand a little bit better, imagine that you have an object called spy. The spy object has the ability to concoct fantastic schemes to take over the world.

So if another object, say, the Control object, messages spy with a message to concoct a new scheme:

```
[spy concoctScheme];
```

then spy might do something like

```
- (void) concoctScheme
{
    [currentScheme autorelease];
    currentScheme = [ [ Scheme alloc ] init ];
    return;
}
```

The rule is that if you create an object, you are responsible for releasing it. By sending currentScheme an autorelease message before returning it, spy is declaring that currentScheme does not need to exist beyond the current scope. By the way, the above example also illustrates nesting of messages, which works in a similar way to nesting function calls.

What if the spy had already passed the currentScheme object to its trusted partner, dirtyCounterSpy, and then subsequently sent the autorelease message to currentScheme as above? Eventually the other object might message the old currentScheme, which has probably long since been auto-released, and the dirtyCounterSpy object would crash hard, taking Control and spy and all the other objects in the secret espionage program with it.

The way to solve that would be for dirtyCounterSpy to send the older currentScheme object a retain message. Sending a retain message increments the older currentScheme's reference count in the autorelease pool. That way when the spy object sends an "autorelease" message to currentScheme, the reference count drops only to 1 instead of 0. The dirtyCounterSpy object can now keep the currentScheme object even though the object has been released by the spy object, and, in fact, even if the spy object is destroyed the dirtyCounterSpy object can use "retain" to keep currentScheme.

I hope this dirty lesson in in the affairs of espionage has briefly illustrated reference counting, and perhaps in other, more exciting ways it's illustrated a little of the power of OPENSTEP. There's quite a bit more to Foundation Kit, mostly in the area of abstraction, that is somewhat beyond the scope of this article. You have what you need, however, to grab a copy of Rhapsody and create that killer app. MT

TestTrack

Your Total Bug Tracking Solution

Powerful enough to handle all your bug tracking needs. **Affordable** enough to put on everyone's desktop.

Introducing TestTrack 1.5 and Solo Bug!

Discover the tool more and more of today's top software developers are using to improve the quality of their Mac and Windows applications—**TestTrack**.

- Track bugs, feature requests, customer information, and more.
- Use on Mac or Windows 95/NT platforms.
- Produce concise reports.
- Link engineers, testers, managers, even tech writers.
- Automatically route bugs to team members.
- Save time and improve tech support by giving **Solo Bug**, our stand-alone bug reporter, to your customers (comes free with TestTrack 1.5).
- Track the history of each bug.
- Multiple users, full security!
- Ideal for consultants too!

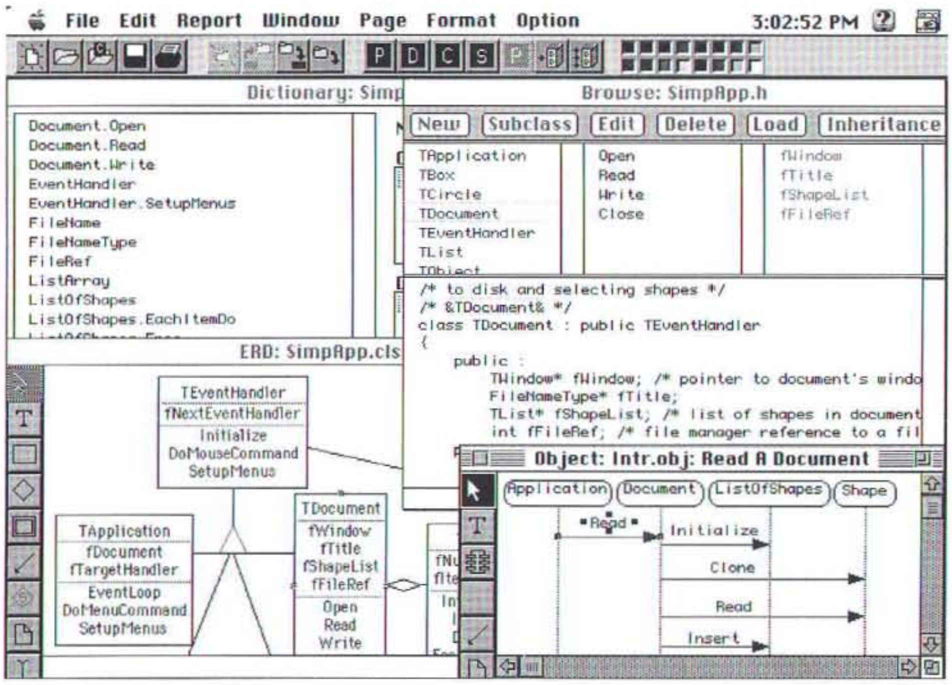
To order call 513-683-6456

Seapine Software, Inc. 1066 Seapine Ct. Maineville, OH 45039

sales@seapine.com
http://www.seapine.com

Seapine Software

Visit MacTech Magazine's Web site!
http://www.mactech.com



```
Document.Open
Document.Read
Document.Write
EventHandler
EventHandler.SetupMenus
FileName
FileNameType
FileRef
ListArray
ListOfShapes
ListOfShapes.EachItemDo
```

```
TApplication    Open     fWindow
TBox            Read     fTitle
TCircle         Write    fShapeList
TDocument       Close    fFileRef
TEventHandler
TList
```

```
/* to disk and selecting shapes */
/* &TDocument& */
class TDocument : public TEventHandler
{
    public :
        TWindow* fWindow; /* pointer to document's windo
        FileNameType* fTitle;
        TList* fShapeList; /* list of shapes in document
        int fFileRef; /* file manager reference to a fil
```

The sporty new interface in MacA&D 6.0 enables designers to click to properties, dictionary details, integrated specifications, code or related requirements for any selected diagram object.

See http://www.excelsoftware.com for more details.

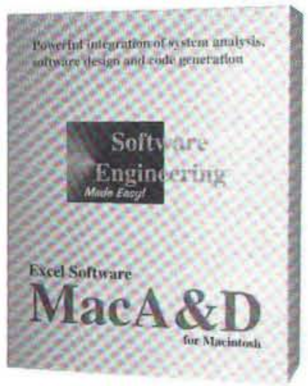
Software Engineering Made Easy!

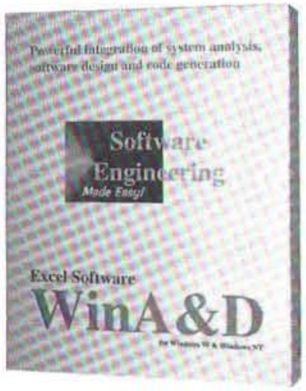
MacA&D and WinA&D simplify system analysis, track requirements specifications, automate popular modeling techniques and generate code from your design. Throughout the process, extensive verification reports catch errors early to help you produce a quality product.

Powerful capabilities are just one advantage. Years of innovation have made these products easy to use and each includes extensive examples and tutorials. MacA&D and WinA&D share documents so your project team can use Macintosh, Solaris, HP-UX or Windows 95/NT computers. Select from OMT, Booch, Shlaer/Mellor, Coad/Yourdon, Fusion or Jacobson methods or pick and mix the best of each.

- Structured Analysis & Design
- Object-Oriented Analysis & Design
- Data, State and Task Modeling
- Integrated Code Editing & Browsing
- Multi-User Dictionary
- Requirements/Use Cases
- Generates C, C++, Delphi, SQL, etc.

Powerful Tools For Software Design


Powerful integration of system analysis, software design and code generation
Software Engineering
Made Easy!
Excel Software
MacA&D
for Macintosh


Powerful integration of system analysis, software design and code generation
Software Engineering
Made Easy!
Excel Software
WinA&D

MacA&D
for Macintosh, Solaris and HP-UX

WinA&D
for Windows 95/NT

See our web site or drop us an email to receive detailed technical brochures for MacA&D or WinA&D. Check out the new MacTranslator and WinTranslator products to generate design documents automatically from existing source code.

Excel Software
515-752-5359

by Michael Rutman, independent consultant

Building an OPENSTEP Application

Is it really as easy to program in OPENSTEP as they say?

OPENSTEP AND NEXTSTEP

In 1985, Steve Jobs left Apple and formed NeXT. He found the latest technologies and brought together a team of developers to turn the newest theories into realities. To verify the new technologies, during development Steve would often stop development and have the entire team use the system they were creating. Several existing apps were created during these day or week long kitchens. More importantly, each developer knew how the framework would be used. If developers had a hard time using an object during a kitchen, they knew they had to rework that object.

The system they created was called NEXTSTEP. NEXTSTEP has several layers, and each layer was state of the art in 1985. In the 12 years since Steve picked these technologies, the state of the art may have moved, but not advanced.

The underlying OS is a Mach mini-kernel running a UNIX emulator. For practical purposes, Mach is a flavor of BSD UNIX. However, the Mach mini-kernel offers programmers a rich set of functionality beyond what UNIX provides. Most of the functionality of Mach is wrapped into the NEXTSTEP framework, so programmers get the power and flexibility of Mach with the ease of use of NEXTSTEP.

Sitting on top of Mach is Display Postscript. Postscript, the language of printers, is a nice graphics language. NeXT and Adobe optimized Postscript for displaying on the screen and produced a speedy and powerful display system. NeXT's framework hides most of the Postscript, but if a programmer wants to get into the guts, there are hooks, called pswraps, to work at the Postscript level.

NEXTSTEP needed a language, and deciding on a language was difficult. In 1985, there were a lot of theoretical languages that claimed to be Object-Oriented, but few were in actual use. C++ was not shipping, but Objective-C was. Furthermore, Objective-C was a much easier language to learn and use. Unfortunately, the rest of the industry went with C++. NeXT responded by adding C++ to Objective-C. Today, under OPENSTEP, programmers can use Objective-C, Java and C++. However, the framework is still based on Objective-C, so learning Objective-C will be important.

Using Objective-C, NeXT created the AppKit. The AppKit is a framework for creating applications. There are many frameworks for making applications today, but the original AppKit is the easiest to use. In addition to the AppKit, NeXT expanded with the Music Kit, Foundation Kit, and EOF (Database Kit). With these kits, a developer can create full featured applications very quickly.

The last layer of NEXTSTEP is the tools. NeXT has implemented a very nice suite of developer tools. NeXT has separate applications for each part of development, but the applications communicate with each other providing an integrated environment. Source files are edited in Edit.app, Projects are maintained and compiled in ProjectBuilder.app, Interfaces are created and edited in InterfaceBuilder.app, and debugging is done with GDB from Edit.app. In addition, there are command line alternatives for any of these GUI applications. Each of these applications work with the command line

Michael Rutman is a software developer with experience developing for several platforms, including Macintosh, NEXTSTEP, Newton, Pilot, and Windows NT. While working at Software Ventures, he lead the development of Snatcher and MicroPhone Pro for NEXTSTEP. He also worked on the MicroPhone Pro for Macintosh product line. He now works as an independent consultant on a variety of projects including encryption, compilers, web based add-rotation software, and ship stevedoring. To contact Michael Rutman send mail to moose@manicmoose.com.

programs, so going between the two is painless. Personally, I spend most of my time in the GUI applications, but I know other programmers who use emacs exclusively.

Now with NEXTSTEP defined, what is OPENSTEP? OPENSTEP is the same as NEXTSTEP, but does not include the development tools or the OS. OPENSTEP will run under any OS, and any development tools can be used to create OPENSTEP applications. At least that is the theory — we will see what Apple decides for Rhapsody. At the time of writing this article, it looks like Apple's Rhapsody will be close to NEXTSTEP.

OPENSTEP comes with tutorials to explain exactly how to use the development tools, so I will gloss over the details where they are straight-forward. I will create a networked lunch application using NEXTSTEP 3.3, which will be similar to OPENSTEP, but not quite. Each person on the network will be able to say where they want to eat, and everyone else running the application will see their choices. In general, development under NEXTSTEP is identical to development under OPENSTEP.

Creating an Application

All OPENSTEP applications start with ProjectBuilder.app. After launching ProjectBuilder.app, we create a new application called "lunch." ProjectBuilder also lets you create bundles, palettes, tools, libraries, and more. Starting source files and interfaces are automatically created and placed in ProjectBuilder. **Figure 1** shows ProjectBuilder with a new application. Double-clicking lunch.nib will open the interface file for the lunch application in InterfaceBuilder.

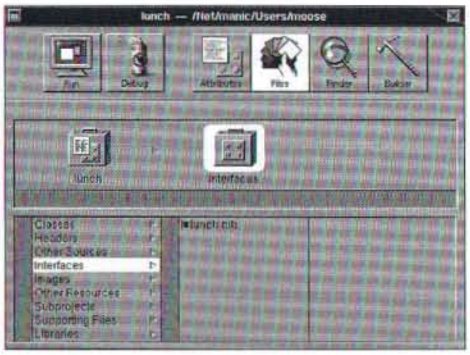
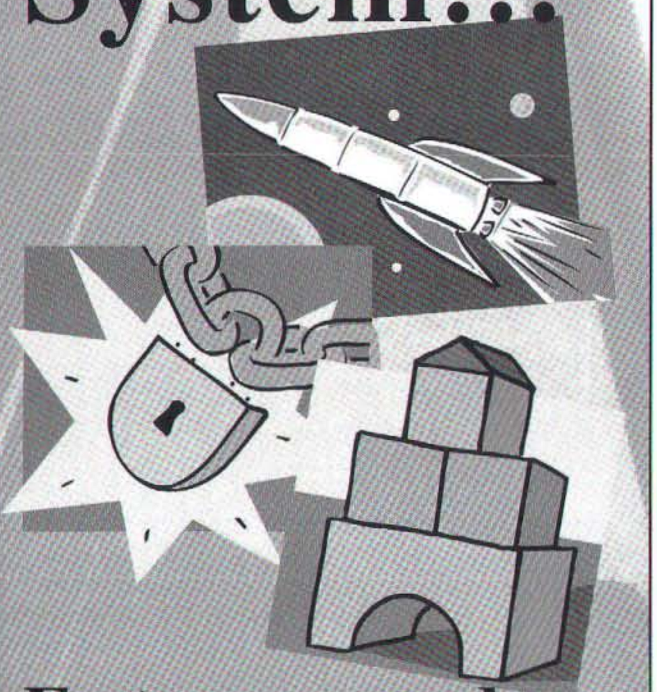


Figure 1.

InterfaceBuilder creates and edits the user interfaces for OPENSTEP applications. The interfaces are stored in "nib" files. Buttons, windows, lists, and other UI elements can be added to an application graphically. Some of these steps are hard to explain if you don't have OPENSTEP in front of you, but once you have OPENSTEP, you will see that this is all straight-forward pointing and clicking.

The lunch application will contain a text field, a button, and a scrolling LunchView object. The text field and button are already provided by InterfaceBuiler and can be dragged into the window from the palette. The LunchView object is a placed by dragging a CustomView and resizing it. InterfaceBuilder's menu item called Group In ScrollView moves the custom view into a scrollview.

The dtF Database System...



Fast, secure and easy-to-use !

dtF is a high-performance SQL database engine supporting MPW C/C++, Symantec C/C++, Metrowerks CodeWarrior (all compilers 68K and native PPC). Separate versions available for use with HyperCard 2.x, SuperCard 2.x, SmalltalkAgents, Apple Media Tool and Pictorius Entrada! **dtF** is fully OpenDoc™ compatible. With its unique direct integration technology, **dtF** is fully contained in your application, there is no need for separate drivers, INITs or background processes. Standalone applications built with **dtF** are royalty-free. **dtF** is fully CD-ROMable and has modest runtime requirements, making it an ideal choice for Powerbook or Multimedia applications. **dtF** supports cross-platform development on Windows 3.11, 95, NT and OS/2 with an identical API across all platforms.

dtF The Relational Database System

dtF Americas, Inc.
19672 Stevens Creek Blvd.,
Suite 128
Cupertino, CA 95014, USA

Phone: (800) DTF-1790
Fax: (510) 828-8755
Internet: dtF@interramp.com
http://www.thetagroup.ilk.de

The professional web page development tool

- Award winning spell checker
- True, continuously updated, WYSIWYG preview
- Tables (creating, editing, and importing)
- Limited site management
- Forms
- Page colors
- Frames
- Text and styled text importing
- Add your own tags
- Very customizable
- Preview with different browsers
- Includes two free upgrades

World Wide Web Weaver

MacUser, August 1996

SiteWeaver

- Check/Edit/Update all of your links on your site on seconds!
- Upload pages and sites directly to the web
- Work offline and then let SiteWeaver™ synchronize your work later
- Easy Mac interface for novices or full time professionals
- Works with just about any graphics/plug-in program, page editor* and web browser

For the latest ordering information point your web browser to http://www.MiracleInc.com or call (315) 265-0930.

InterfaceBuilder has an Inspector window for setting attributes on objects. The inspector is used to fine tune the locations of the controls. The inspector also allows us to disable the Eat button. The resulting window is shown in **Figure 2**.

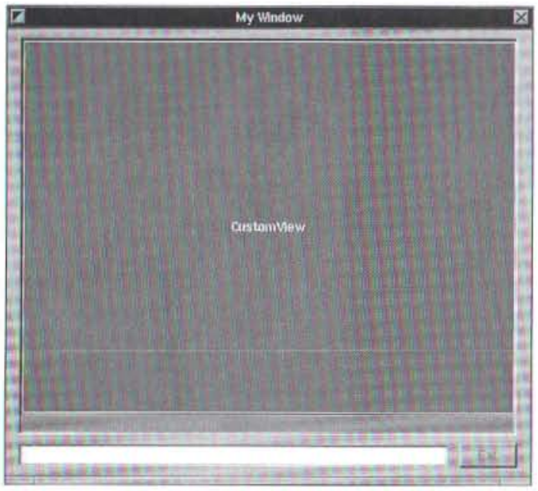


Figure 2.

User Interface Control

The lunch application uses two custom objects. InterfaceBuilder allows custom objects to be created and automatically integrated into ProjectBuilder. When a nib is opened, a window titled with the name of the nib appears in the lower left. This window lists all the non-graphical objects in the nib file, as well as the windows. Custom objects are created under the classes tag.

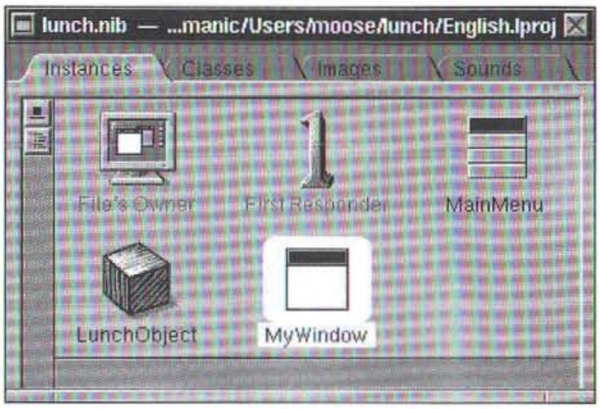


Figure 3.

Every nib file is owned by an object, which is typically the object created just before the nib is loaded to hook the nib into the application. This object is represented by the File's Owner icon in **Figure 3**, and can be set in the Inspector. For lunch.nib, the File's Owner is an Application object because this is the starting nib. A subclass of the Application could be made the File's Owner, but delgation works better.

Some objects, such as Window, Text, and Application, have hooks built into them to delegate functionality to other objects. We will create a custom object to act as our Application's delegate. Delegates modify the functionality of core objects by implementing methods. They need only implement some of the methods. If a method is not implemented, then the core object will know not to call it. Objective-C allows run-time checking of what routines are implemented, so delegates can be very light-weight objects. The LunchObject is the delegate to both the Application and a text field.

Custom objects are created by selecting Classes from the tags. The lower left window will now look like **Figure 4**. In the bottom of the window is a pull-down menu called Operations. This menu contains the operators used to create custom objects. The subclass operator will create a subclass of Object. The new object starts with the name MyObject, but should be changed to LunchObject for this project. LunchObject will implement methods for controlling the nib's custom interface objects. Outlets in the LunchObject are required for the window, edit button, lunch view and text field. Create 4 outlets and name them window, lunchField, lunchView, and lunchButton. Also create an action called eat. The unparse operator will create a template, and puts the source files in ProjectBuilder. Selecting the instantiate operator will create an instance of LunchObject.

Apple Developer University Has New Classes

- *We've updated our QuickTime VR training for version 2.0.*
- *Our AppleScript class now includes creating CGI applications for your Mac OS web server.*
- *New course: Programming Java on the Macintosh*
- *Also note our new location in Boston, Massachusetts*

Classroom

Check our website for complete descriptions, schedule and fees: http://devworld.apple.com/du
To register call • (800) 282-2732 (U.S.) • (800) 637-0029 (Canada)
• (716) 871-6555 (Outside U.S./Canada)

Self Paced Training

Virtual Tutor for QuickTime VR, $79.95 (part number R01717Z/A)
And many other titles. To order, call the *Apple Developer Catalog* at (800) 282-2732

Online Training

Free online training is available on many popular topics. See Virtual Tutor for QuickTime VR at our website: http://www.devworld.apple.com/du in the online training area.

DEVELOPER UNIVERSITY

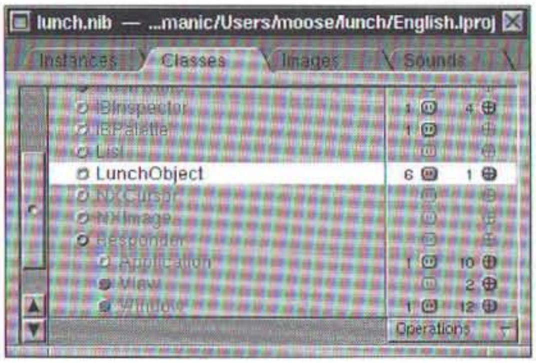


Figure 4.

Holding down the control key, drag from the File's Owner field to the LunchObject. The inspector shows what connections are possible. Selecting the delegate outlet and clicking OK in the Inspector hooks these objects together. No code change is necessary to hook objects together. Dragging from the LunchObject to the window, text field, custom view, and button allows us to set the LunchObjects outlets. To set the buttons action, drag from the button to the LunchObject and set the targets action to be eat.

To enable and disable the Eat button depending on if there is text in the text field, make the LunchObject the delegate of the text field. (One object can be the delegate to many other objects.) Drag from the text field to the LunchObject to set the delegate. Another nice feature for users is to be able to press return in the text field and have the Eat button hit. This is done by dragging from the text field to the button and selecting target:performClick. Again, no code changes are necessary.

Whenever text is changed, we must enabled or disable the button. The delegate method that best handles this is -textDidGetKeys:isEmpty, implemented in Listing 1. This method is called whenever the text field receives keys. For performance reasons, it is not called each time a key is pressed, but for our purposes it will work very well.

```
Listing 1: User Interface code
- textDidGetKeys:sender isEmpty:(BOOL)flag
{
   [lunchButton setEnabled:!flag];
   return self;
}
```

The second custom object is our LunchView. LunchView is created very much like the LunchObject, but it is a subclass of View. It has no outlets or actions, but is unparsed like LunchObject. Instead of instantiating a copy, we click the CustomView we created earlier and use the inspector to set the custom view to be our LunchView.

Once the nib is saved, the user interface is done.

MkLinuX

Microkernel Linux for the Power Macintosh

Why wait for Rhapsody? MkLinux is a complete system, based on Linux 2 and the Mach 3 microkernel. It includes a complete software development system (gcc, gdb, perl, ...), X11R6.3, and hundreds of other commands. Get MkLinux and start working right now with Mach, Objective-C, UNIX development tools, and more...

MkLinux builds and runs most Intel-based Linux software. It works efficiently and reliably on a wide range of Power Macintosh platforms, with other ports on the way. MkLinux has an active user community and a substantial Reference Release, both sponsored by Apple Computer. If you want to get a jump start on Rhapsody, this is a great way to do it!

Visit Apple's MkLinux web site (www.mklinux.apple.com) and Prime Time Freeware's web site (www.ptf.com) to find out more about MkLinux and other fine freeware products.

Prime Time Freeware	info@ptf.com
370 Altair Way, #150	(408) 433-9662
Sunnyvale, CA 94086	(408) 433-0727 fax

NETWORKING

In Objective-C, run-time binding allows two objects with completely different inheritences to implement the same method. However, this flexibility has some costs. Run-time lookups take slightly longer, the compiler cannot catch bad calls, and the caller has to wait to see if the receiver is going to respond. Objective-C partially solves these problems by implementing protocols. A protocol is a list of methods that objects must implement. There is still a cost for run-time lookups, but the compiler will catch bad calls, and the caller can know if a value will be returned. With networking, not having to wait for a response can make a big difference. Our LunchObject supports two protocols, LunchClientMethods and LunchServerMethods.

Inter-Process communication is very easy under OPENSTEP. The first step is forming a connection between two processes. LunchObject is the delegate of Application and gets an **appDidInit** call when the application is done launching. Likewise, it gets an **appWillTerminate** call before the application quits. If a connection breaks, **senderIsInvalid** will be called. These three routines are in Listing 2.

```
Listing 2: Network connection
- appDidInit:sender
{
  remoteHub =
    [NXConnection connectToName:"Lunch" onHost:"*"];
  if (remoteHub)
    {
    //We will be a client
    ourServer  = [remoteHub connectionForProxy];
    [remoteHub
      setProtocolForProxy:@protocol(LunchServerMethods)];
    isServer   = NO;
    }
  else
    {
    //We are first to launch, so we are the server
    ourServer =
      [NXConnection registerRoot: self withName:"Lunch"];
    remoteHub  = self;
    isServer   = YES;
    clientList = [[List alloc] init];
    }

  [ourServer registerForInvalidationNotification:self];
  [ourServer runFromAppKit];
  [remoteHub addClient:self];
  [self updateEveryone];

  return self;
}

- appWillTerminate:sender
{
  [remoteHub removeClient:self];
  return self;
}

- senderIsInvalid:sender
{
  if ( sender != self )
    {
    //server died, we all have to go away
    remoteHub = nil;
    [NXApp terminate:self];
    }
else if ( isServer )
  [self removeClient:sender];

  if ( isServer )
    [self updateEveryone];

  return self;
}
```

LunchObject will act as both server and client. Each LunchObject checks the network for any other LunchObject, if it doesn't find one, then it will register itself as the server. By checking the host *, we check all machines on the network. If we knew a server was running on one particular machine, then we could specify that machine as the host.

Registering self as the root object will cause this object to be distributed over the network. The next LunchObject that runs will check the network, find the first LunchObject registered, and receive a proxy to the first LunchObject. A proxy is an object that pretends to be another object. Any calls to the proxy go through Mach messages to the real object. The real object can return any data it wants, including other objects. New objects can be passed by copy or proxy. By default, the first object will send a proxy to the second object.

Once the two sides are communicating, the client tells the server about itself by asking the server to **addClient** on itself. It is passing a proxy of itself to the server. If a client quits, then it will remove itself from the server. The connection is also set to understand the LunchServerMethods protocol to avoid round-trip calls when possible.

Once the two sides are communicating, they must update all the clients. Listing 3 has the code for adding and removing clients,

The number one AppleScript environment

w i d e n s   t h e   l e a d.

**For professionals,
for novices,
for webmasters,
for solutions providers:**

There's only one serious choice.

**Always the most powerful,
the most feature-packed,
and at the same time,
the easiest to use.**

And it just got better.

**It's already the only tool that lets
you watch local variables, trace
variables, or change values and
fix problems while stepping.**

Now, Scripter hits version 2.0,

with still more unique features:

Handler debugging: Debug handlers without modifying your scripts.

Applet and CGI simulation: Interactively debug live interapplication messages sent to a script application or AppleScript CGI from an applet, your Web server or FaceSpan 2.x. You won't have to modify your Web pages or your CGIs.

Live editing: Debug script applications and CGIs in their normal environment.

Object Map: Explore a graphical view of the class hierarchy of scriptable applications.

Plus: Construct generators; associated terminology; search backwards; faster searching; more navigation tools; makeup of AppleScript strings; fast compile (without debugging); even more enhanced trace log; and more.

Integrated object database: ScriptBase, separately $59, is now included with Scripter. Use it to store your data and media elements (frequently used values, scripts, text, pictures, HTML, headers, file references) and share them between scripts. Manage ScriptBase directly from Scripter.

Scripter® 2.0 with ScriptBASE™

Buy online at
www.mainevent.com
or order, toll-free
1 800 616 8320
info
+1 202 298 9595
email
info@mainevent.com

If you're an AppleScript novice, Scripter will take you by the hand and show you the correct syntax for your statements. If you know what you're doing, Scripter will help you do it faster than any other editor. And Scripter is the only tool on the market with *interactive debugging:* You catch the offending code in the act. Only Scripter lets you fix the problem and continue debugging!

We designed Scripter for productivity and user-friendliness. People come to us for Scripter's industrial-strength debugger, but what really impresses them is the speed and ease at which Scripter allows them to work. It's been used to build major corporate process automation systems.

Scripter offers power-assisted statement construction. Vocabulary access in a single mouseclick.

Multifunction find and replace. A variable watcher and expression evaluator. Tools to change variable values or try out commands, in context, in the middle of debugging. Automatic navigation to subroutines. Background processing. An enhanced trace log. Extensive Drag and Drop support. We could go on, but we just don't have the space here. So get a copy today, and start scripting with power!

Main Event
The Macintosh Scripting Company

Design: Boris Levitin

◆ SUNOS ◆ SOLARIS ◆ MACOS ◆ BEOS ◆ LINUX ◆ FREEBSD ◆ SGI ◆

SINIX ◆ NCR ◆ OS/2 ◆ DEC/UNIX

HP-UX ◆ BSDI ◆ SEQUENT ◆ QNX

PERFORCE®

Fast Source Code Management On Networks Everywhere

◆ WINDOWS95 ◆ INTEL NT ◆ ALPHA NT ◆ POWERPC NT ◆ AIX ◆ SCO ◆

... find us fast... on MacOS, BeOS and everywhere else... f ...

FREE! on the Be CD

PERFORCE SOFTWARE, INC.

1.510.865.8720 **HTTP://WWW.PERFORCE.COM** **INFO@PERFORCE.COM**

as well as the code for updating all the clients. One of the drawbacks of Inter-Process Communication is dead-locking. If one method calls another method on the other side, and that method has a callback to the first side, the two sides could be blocked waiting for the other to complete. Using perform:afterDelay methods prevent this. The perform:afterDelay will wait until the next time through the event loop and perform that method. The original calls will then not dead-lock. OPENSTEP's distributed objects are not too sensitive to dead-locking, but it is a good habit to always protect against it.

```
Listing 3: Client-server communications
- updateEveryoneAfterDelay
{
  [clientList makeObjectsPerform:@selector(updateStats)
    with:nil];
  return self;
}

- (void)updateEveryone
{
  [self perform:@selector(updateEveryone)
    with:nil afterDelay:1 cancelPrevious:YES];
  return;
}

- (void)addClient:client;
{
  [clientList addObjectIfAbsent:client];
  [self updateEveryone];
}

- (void)removeClient:client;
{
  [clientList removeObject:client];
  [self updateEveryone];
}

- (List *)clientList
{
  return clientList;
}
- (void)updateStats;
{
  [self perform:@selector(updateStatsAfterDelay) with:nil
afterDelay:1 cancelPrevious:YES];
}
```

Whenever a client is added or removed, the server calls its updateEveryone method. That method will wait until the next event loop, call updateEveryoneAfterDelay. updateEveryoneAfterDelay calls updateStats in every client, which will call each clients updateStatsAfterDelay after a delay. updateStatsAfterDelay will rebuild everything. Whenever any client changes, it can call updateEveryone in the server and every client will update.

The last three routines needed for networking are to send the actual data. The eat: method saves the user's choice and updates all the clients. Each client's updateStatsAfterDelay method must find out where each user wants to eat by calling wantToEatWhere method. In our code, the updateStatsNow only passes the list of clients to the LunchView, the LunchView will call wantToEatWhere. These 3 routines are listed in Listing 4.

```
Listing 4
- eat:sender
{
  [where release];
  where = [NSString stringWithCString:[(TextField
*)lunchField stringValue]];
  [remoteHub updateEveryone];
  return self;
}
```

```
- updateStatsAfterDelay
{
  List *serverClientList;
  serverClientList = [remoteHub clientList];
  [lunchView updateLunchList:serverClientList];
  return self;
}

- (NSString *)wantToEatWhere;
{
  return where;
}
```

The choice of where to eat is stored in an NSString. NSString is part of the Foundation Kit instead of the AppKit. Intermingling different kits is usually transparant. Foundation Kit objects provide garbage collection. Any object can be set to be auto-released and the next iteration of the event loop will free all the objects in the pool. NSString returns a char * that will automatically be freed later.

LunchView

Displaying the lunch results is done in the LunchView. LunchView is a subclass of view and as such, knows about setting up the display system. The actual code to display is simple, but turning the distributed list into something easy to display is a bit more complicated. In C++, there would probably be 3 or 4 objects to do this, but in our example we will have only one helper object, LunchChoice. Objective-C objects tend to be larger than C++ objects. All functionality for viewing should be placed in the view object.

LunchChoice, our helper object, will store one restaurant choice and the number of times this restaurant has been requested. The code is in Listings 5 and 6. The LunchChoice stores the restaurant in an NSString, which will take care of garbage collecting for us. The + sign before some of the methods are the same as C++ static methods, but are called **factory** methods. LunchChoice uses these factory methods to store the MaxValue so LunchView can scale appropriately.

NEXTSTEP has a convention of using init... for initialaztion methods and free for freeing methods. OPENSTEP has a convention for releasing objects instead of freeing them. Releasing, rather than freeing, an object is used by the garbage collection system. The initLocation method will call [super init], then an NSString object. The free method is called whenever we are done with this object, and it releases the NSString allocatted as well as this object. The rest of the methods are straight-forward.

```
Listing 5: Lunch Choice Header
#import <appkit/appkit.h>
#import <foundation/foundation.h>

@interface LunchChoice:Object
{
  NSString *where;
  int      value;
}

+ ResetMaxValue;
+ (int)MaxValue;
- initLocation:(const char *)location;
- incrementValue;
- (NSString *)location;
- (int)value;
- free;
@end
```

Power Mac and Macintosh Developers:

FIND OUT FAST WHAT'S GOING ON IN MEMORY

THE MEMORY MINE™

- See memory allocation in any open heap at a glance.
- Easily spot memory leaks.
- Flags heap corruption when it happens.
- Works with source level debuggers to let you find memory problems fast.
- Stress applications on the fly with Purge, Compact, and Zap.
- Allocate memory at will for precise stress testing.
- Log heap data - easily document heap status over time.
- No need for source code: nothing inserted in code; no patches to the system.
- Works with 24-bit, 32-bit, and modern memory managers.

For Macs with 68020 or better. Requires System 7.0 or later.

only $99 US
Order now from Adianta, Inc.
Phone: (415)781-8052 • FAX: (415)781-8053
AOL:Adianta • Internet:adianta@aol.com
For VISA, MC, or American Express orders by mail, fax, or Internet, please include name, address, card number, expiration date, and phone number or email address.
Also available through APDA and the MacTech Mail Order Store.

for more information contact

Adianta, Inc. • 582 Market St. #911 • San Francisco, CA 94104

```
Listing 6: LunchChoice Source
#import "LunchChoice.h"

@implementation LunchChoice

static int MaxValue = 0;

+ ResetMaxValue
{
  MaxValue = 0;
  return self;
}

+ (int)MaxValue
{
  return MaxValue;
}

- initLocation:(const char *)location
{
  [super init];
  where = [NSString stringWithCString:location];
  value = 0;
  return self;
}

- incrementValue:
{
  value++;
  if ( value > MaxValue )
    MaxValue = value;
  return self;
}

- (NSString *)location;
{
  return where;
}
```

```
- (int)value;
{
  return value;
}

- free;
{
  [where release];
  return [super free];
}

@end
```

LunchView, being a subclass of view, has a standard initialization method called initFrame. initFrame will call [super init] and create storage for the list of choices. The free method will free all memory associated with LunchView. Unlike C++ with constructors and destructors, init and free methods must be manually called. These routines are in Listing 7.

```
Listing 7: LunchView initialization and free methods
- initFrame:(const NXRect *)frm;
{
  [super initFrame:frm];
  choices = [[List alloc] init];
  return self;
}
- free
{
  [choices freeObjects];
  [choices free];
  return [super free];
}
```

LunchView's updateLunchList method gets called whenever there is a change in restaurant. The list passed is located in the server process, but by this point, it is transparant to LunchView. Any call to the list passed in will act like a call to a local object, but be a bit slower. This routine deletes the existing list and recreates a new list from scratch. Not the most efficient algorithms, but works for our purposes. The updateLunchList method calls the getChoice method with a string value. The getChoice method will return a LunchChoice object of that name, creating one if necessary. Once a LunchChoice is found, it's value is incremented. When the new list has been parsed, the view is resized and redrawn. The list handling routines are in Listing 8.

```
Listing 8: LunchView's choice handling methods
- (LunchChoice *)getChoice:(const char *)location
{
      int       count;
  const char    *thisLocation;
      LunchChoice*thisChoice;

  for ( count = [choices count] - 1; count >= 0; count-- )
    {
    thisChoice   = [choices objectAt:count];
    thisLocation = [[thisChoice location] cString];
    if ( !strcmp( thisLocation, location ) )
      break;
    }
  if ( count < 0 )
    {
    thisChoice   =
      [[LunchChoice alloc] initLocation:location];
    [choices addObject:thisChoice];
    }
  return thisChoice;
}

- updateLunchList:lunchList;
{
  int         count;
  NSString    *thisLocation;
  LunchObject  *thisHandler;
  LunchChoice  *thisChoice;

  [choices freeObjects];
  [LunchChoice ResetMaxValue];
  for (count = [lunchList count] - 1; count >= 0; count--)
    {
    thisHandler  = [lunchList objectAt:count];
    thisLocation = [thisHandler wantToEatWhere];
    thisChoice   = [self getChoice:[thisLocation cString]];
    [thisChoice incrementValue];
    }

  [self updateFrameSize:[choices count]];
  [self display];
  return self;
}
```

Frame resizing is done in the updateFrameSize method. When a view in a scrolling view is resized, everything works correctly. No extra code is needed. However, code to make sure the view doesn't shrink smaller than the visible area is necessary. Otherwise, there will be areas of the ScrollView that are not cleared correctly. This routine is in Listing 9.

```
Listing 9: Frame handling
- updateFrameSize:(int)numberOfColumns
{
  NXRect      theFrame, visibleBounds;

  [self getFrame:&theFrame];
  [superview getVisibleRect:&visibleBounds];

  if (visibleBounds.size.width
      < numberOfColumns * kWidthPerColumn)
    theFrame.size.width = numberOfColumns * kWidthPerColumn;
  else
    theFrame.size.width = visibleBounds.size.width;
  [self setFrame:&theFrame];
  return self;
}
```

The drawSelf method is responsible for all drawing. In general, the rectangle to be updated is passed as a pointer to this method and a rectCount of 1. However, for optimization, it is possible to pass an additional 2 rects to specify a non-rectangular area to update with a rectCount of 3. The first rectangule, however, must always contain a rect that can update the entire window.

When drawSelf is called the postscript context has been set for the drawing. Postscript commands sent to the postscript server will be processed and displayed. There is a buffering built in, so there will be little to no flicker. If there is any flicker, disabling the flushing is a single call. Turning the flushing back on and flushing the window will blast the pixels in one shot. LunchView did not need this extra bit of code.

The first step is to set the font and color LunchView is going to use. The font chosen is a screenFont, but if we want to print this window, OPENSTEP will automatically switch to a printer font. Next, a rectangle to draw each choice is created. Finally, the choices are iterated and information is displayed. The NXRectFill shows a weighted value for each choice and the Psshow displays the choice. Listing 10 has the source to drawSelf and Figure 3 has the final screenshot.

```
Listing 10:
- drawSelf:(const NXRect *)rects :(int)rectCount;
{
  int    count;
  NXRect thisRect;
  PSsetgray( NX_BLACK );
  [[[Font newFont:"Helvetica"
    size:10.0 matrix:NX_IDENTITYMATRIX] screenFont] set];
  thisRect.origin.y = bounds.origin.y + kOffset*2;
  thisRect.origin.x = bounds.origin.x + kOffset;
  thisRect.size.width = thisRect.size.width + kWidthPerColumn
- kOffset*2;

  NXEraseRect( rects );
  for ( count = [choices count] - 1; count >= 0; count- )
    {
      LunchChoice*thisChoice = [choices objectAt:count];

      PSmoveto(  thisRect.origin.x + kOffset,
            bounds.origin.y + kOffset );
      PSshow( [[thisChoice location] cString] );
      thisRect.size.height =
          (bounds.size.height - kOffset*3)*
            ((float)[thisChoice value])/
              ((float)[LunchChoice MaxValue]);
      NXRectFill( &thisRect );
      thisRect.origin.x += kWidthPerColumn;
    }

  return self;
}
```

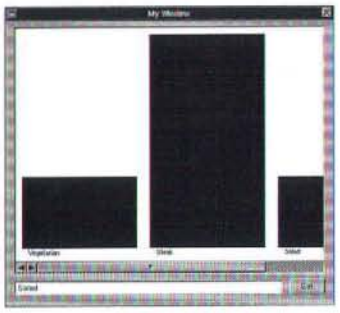


Figure 5.

What else can I do that is easy?

From Interface Builder, printing and spell checking can be added with no coding at all. Setting a font would require an additional method, but roughly only 5 lines of code would be added or changed.

Setting up a chat line so people can talk while this is going on wouldn't be difficult at all. The list of LunchObjects is already distributed, so a list of chat objects can be easily added.

Perhaps statistics should be kept. OPENSTEP provides an NXStringTable object that allows storage to a file of a key value pair. The NXStringTable object makes simple hashing a dream.

When marketing droids say that developing in OPENSTEP is 3 times as fast as conventional development, everyone doubts them. From personal experience, I've found it to be an almost 10-fold increase. This entire application, including complete networking code, took me less than 4 hours to write. How long would it take to write this same application on a Macintosh or under Windows? MT

Visit MacTech Magazine's Web site!
http://www.mactech.com

Name Your Poison

Java OpenDoc (ODF) Tools Plus MPW PowerPlant Roaster CodeWarrior C++ Procedural C TCL SYMANTEC OOFILE neoAccess

AppMaker supports most popular languages and frameworks. Just point and click to design your user interface, then let AppMaker create resources and generate "human, professional quality code" to implement your design.

Use AppMaker as a prototyping and productivity tool or use it as a learning tool for Mac programming techniques or for new environments such as OpenDoc, Java, or PowerPlant.

AppMaker is just $299 and includes a one-year subscription on CD.

B•O•W•E•R•S Development

603-863-0945
bowersdev@aol.com
http://members.aol.com/bowersdev

by Edward Ringel

OpenDialog

Specialized dialog functions library for the industrial strength programmer

The Dialog Manager is a flexible package that is used by almost everyone creating forms for Macintosh applications. For some applications, these forms are limited to setting preferences or a few other limited tasks. In this situation, extensive customization of the dialog is unnecessary, and Toolbox calls are more than sufficient. In other circumstances, a series of dialogs, modal and modeless, may be the core of the project and represent the bulk of the user interface. In these applications, customization of the dialogs may be desirable and even mandatory. In this setting, the direct Dialog Manager calls may be clumsy and require large amounts of coding to achieve input filtering, relating data structures to items, cursor management, and the like.

Object oriented technologies have devised schemes to exercise considerable control over forms-style GUIs. Procedural programmers usually have a collection of routines and libraries tucked away which simplify the problem, at least to a degree. From time to time various programming tools become available which also purport to simplify the life of the dialog-forms programmer. OpenDialog is such a package.

Dialog Manager Replacement

OpenDialog is engineered and supported by FGM, Inc. of Herndon, VA. This company's primary business activity is writing end-user software, not creating programming tools. Nonetheless, their collection of routines and their systematic approach to addressing Dialog Manager problems and deficiencies resulted in a library that clearly has commercial appeal. As FGM states in its ads and introductory literature, this is a replacement for the Dialog Manager. It is recommended that you do not use OpenDialog and Dialog Manager calls on OpenDialog dialogs because, as the manual says, "the results may be unreliable."

When OpenDialog creates a new dialog (modal, moveable modal, and modeless dialogs are all supported) it creates private

Object oriented technologies have devised schemes to exercise considerable control over forms-style GUIs.

data structures and tags the dialog as its own. The library supplies a function, DMX_ItzaODlogWindow(), to test whether a dialog belongs to the OpenDialog manager or not. As a replacement for the Dialog Manager, OpenDialog has an equivalent function for just about all of the Dialog Manager functions.

With the exception of a few calls, the replacement function and the original function have very similar names and parameter lists; this is obviously helpful and important. Parameter lists are different when functionality is extended. DMX_GetNewDialog() may be a good example; this function's parameter list allows the programmer to specify if the dialog is

Ed Ringel is Contributing Editor for product reviews for MacTech Magazine. In his spare time, he is a respiratory and critical care physician in Waterville, Maine. He can be reached at eringel@mint.net.

to be modal or modeless, to assign a text style for the dialog, and to set flags for event handling.

As may be apparent from these two function call examples, the function calls all start with DMX_; this permits easy differentiation from Toolbox calls in your code. Throughout the library, there is a clear and consistent nomenclature that was very easy to follow; this helped shorten the learning curve considerably.

Extensions to the Dialog Manager

Clearly, one does not purchase a library to simply have a different prefix to a function. OpenDialog provides a very wide range of extensions to the services provided by the Toolbox.

One very nice feature of the product is item tagging. Every item in a dialog can have text associated with it; OpenDialog takes advantage of this to allow for customization of the item at the resource level. As I describe below, one powerful use of this feature is to customize an edit field to create filtering. Another is to "name" an item and free the programmer from needing to remember a DITL number. Controls and other elements may also be customized with item tagging. I think that the ability to set filtering (and other services) at the resource level is in the best spirit of Macintosh programming tools.

The largest (and probably for many programmers, the most important) group of services have to do with editable text boxes. By tagging the edit text item's default text in the resource editor, one can limit the number of characters in an entry, force numerical data only, etc. Getting and setting data from a text box is much, much easier than with Toolbox calls. To get any item's text, one simply calls pascal void DMX_GetItemText(const DialogPtr aDlog, short aItemNumber, Str255 aString), and to set the text, one calls the counterpart function, pascal void DMX_SetItemText(const DialogPtr aDlog, short aItemNumber, const Str255 aString). Specialized getters and setters for date, time and latitude and longitude are also supplied. Although one might functionally group cursor control with other types of service extension, the library supplies an automatic I-Beam cursor if the field is initialized correctly. To give a flavor of OpenDialog calls, below is a listing from a short program I converted to OpenDialog. This function creates a modeless dialog, counts the number of dialog items, tests to see if the item is of type editText, and if it is, installs automatic I-Beam management.

```
void CreateModelessDialog(short DialogID, DialogPtr
*theNewDialog)
{
  short  tempType;
  short  numItems;
  short  i;

  *theNewDialog = NULL;
  *theNewDialog = DMX_GetNewDialog
      (DialogID, kDMX_Modeless, 0L,kDMFlag_DoIdleEvents);
  if (*theNewDialog !=NULL) {
    numItems = DMX_CountDITL(*theNewDialog) +1;
    for (i=1;i<numItems; i++){
      tempType = DMX_GetItemType(*theNewDialog,i);
      if (tempType==editText)
        DMX_SetIBarFlag(*theNewDialog,i,true);
      }
    ShowWindow(*theNewDialog);
    SetPort(*theNewDialog);
  }
    return;
}
```

WHY INSTALLER VISE?
ASK METROWERKS.

"Installing CodeWarrior is a herculean task, we ship over 400 megabytes on each CodeWarrior CD release. It would frankly be impossible for us to undertake this task three times a year without Installer VISE from MindVision. VISE is simply a godsend."

–Greg Galanos, President, Metrowerks Corporation

Installer VISE.
The Professionals' Choice.

MindVision Software **www.mindvision.com**

Developer DEPOT™

Want to know what products are available for MacOS development? Check out Developer Depot™ <http://www.devdepot.com>

OpenDialog also supports a variety of extensions to the dialog manager that are not directly edit-field related. Among others, OD has scrollable text, twisters (Finder type triangles), and specialized support for multiple pane dialogs. Although some elements, such as lists, are not supported directly, there is help provided with functions such as DMX_CouldFocus(), which allows the active user item to receive keyboard events.

OpenDialog contains a number of very valuable utility functions. It has a fairly large subset of ANSI library and other string and number utility functions, which eliminate dependence on much of the ANSI library functions. There's a nice call that gets extensive volume information. There are several functions for putting up a quick "question" alert, an error message alert, and a password dialog.

Perhaps of greatest utility are the callback functions. One can attach a callback to *any* item for *any* purpose. (Well, I guess I'm exaggerating a little, but you really can attach a callback to any item.) Event handlers, specialized update routines, data entry filters, cursor callbacks, action procedures and the like can all be attached with relative ease. While this part of the package has the steepest learning curve, it clearly also holds the most bounty. Since much of the interaction is with the Toolbox, many of the callbacks are framed as Universal Procedure Pointers, and appropriate macros are provided. All in all, this is a very robust package.

WHAT OPENDIALOG IS NOT

OpenDialog is not a GUI builder, nor is it an application shell. This is a replacement and extension to the Dialog Manager, and no other services are provided, except in bits and pieces. For example, in my program which I converted to OD, I still had to SetPort() on my own after bringing a modeless window to the front. Window Manager routines work on OpenDialog dialogs without difficulty, and are used extensively when working with modeless dialogs. I think it is particularly important that OpenDialog's limitation be recognized. The novice or early intermediate programmer will certainly be able to use the package, but will feel somewhat unsupported in the lack of event loop support and help with printing, etc.

DOCUMENTATION

The documentation of this package is difficult to characterize, as parts of it are very good, and other parts are problematic. There is an initial short introduction to the OpenDialog programming paradigm, and the remainder of the manual is a description of all the functions. The document, as I received it, was in e-doc format and about 12 months out of date.

The documentation of this package is difficult to characterize, as parts of it are very good, and other parts are problematic.

A large supplement of functions had been added and were in a Simple Text "Read Me" file.

The documentation present is quite good. The function descriptions follow a nice format, and I knew how to use a given function after reading the manual section describing it. The introduction gave me a good general feel for how the guys at FGM were approaching the dialog problem. e-doc format does not permit hypertext cross referencing — I think this is a must for electronic documentation; I would very much have liked to see some means of accessing a function description rapidly. I was unhappy that the manual was dozens of new functions out of date.

The primary instruction for the real-life use of the program is in the sample program and the DMC.h header file. The sample program is large, complex, and somewhat hard to follow; the demo shows off the features nicely, but I would have preferred a series of smaller programs that were easier to read and digest. My test conversion program used modeless dialogs, and there was no demonstration of how to use OpenDialog for modeless dialog work. An email to FGM resolved this quickly, and Charlie Vass, the engineer, was most helpful. When all is said and done, the information is there, and describes the product correctly, but could do with improvement in formatting and accessibility.

The Bottom Line

OpenDialog is a set of powerful routines that can simplify life tremendously for programmers pushing the Dialog Manager to its limits. I could see this product being quite helpful to anyone developing complex electronic forms in a procedural language. There is probably less of a need for this product with object oriented frameworks, given the design paradigm of a window view object populated by other views object. This product should not be used by a novice programmer.

OpenDialog is available directly from FGM for US$ 259. The package consists of documentation, a header file, and both 68K and PPC libraries for CodeWarrior. Libraries are available for the Symantec environment. All functions are declared pascal, and the commercial package comes with a Pascal interface file.

OpenDialog Lite is a free, limited version of OpenDialog (lacks some of the newer functions and does not have any documentation except for the sample program and header file) and is available on the CW 11 disk. It's a great way to test drive the product. I urge anyone with some programming experience and a need for sophisticated forms to look into this package.

Products Reviewed in this Article

OpenDialog, version 1.2.3 using C interface and CW 10.

Usefull URL's

<www.fgm.com>

MT

CRUSHER!™ Data Compression Toolkits

The best high performance, portable compression libraries for DOS, Windows, OS/2, Unix, Macintosh, embedded systems, and practically anything else, period.

New Version 3.2

Robust, 45-Function API
Buffer and File Compression
Portable Archives and Data
Disk Spanning
Encryption
Self-Extracting Executables
On-line Help
Full C Source Code

DOS	$249
Win16	$299
Win32	$299
OS/2	$349
Unix	$349
Macintosh	$349

FREE DEMO

DC Micro Development

Tel 606-245-4175
Fax 606-245-9305
info@dcmicro.com
http://www.dcmicro.com

Call 1-800-775-1073

Help Make MacTech Work

Here at *MacTech Magazine*, we rely heavily on outside writers for most of the material that appears in our pages. If readers did not participate in the magazine, sending us their ideas and taking the time to write articles, there would be no *MacTech*. *MacTech Magazine* is not a staff of writers sending a constant stream of one-way messages outwards; it's a living, evolving network of readers conversing with one another, educating one another, sharing their knowledge, their experience, their interest, their trials and tribulations and joys and successes in the constantly unfolding story of programming the Macintosh. *MacTech Magazine* doesn't just happen: it's what the community makes it. If we carry reports of future trends and technologies, if we teach useful methods, if we review new books and tools, if we provoke thought, provide help, ride the wave of current interests and concerns, it is only because we reflect the thoughts of our readers, who speak through our pages.

You are invited to involve yourself in this exciting conversation amongst readers. No matter who you are, no matter what your credentials may be, if you have a tale to tell, a trick to share, a technique to teach, we want you to consider joining the family of those who write for *MacTech*.

Don't just wait for a topic to be covered or a technique explained in *MacTech!* Take responsibility! Write us an article yourself!

To write for *MacTech*, just send for our Writer's Kit. It's a Microsoft Word file containing the Styles you need to use, and giving lots of helpful advice and information, including all the legal stuff. You can let us know what you're writing about, or, if you want to, you can just write the article and spring it on us when it's done. [**Note:** We also have a need for people willing to make themselves available to write occasional product/book reviews.] If we publish your article, you'll be paid for it!

Write to us, the editorial staff, at editorial@mactech.com (or one of the other addresses listed on page 2 of the magazine). Take the future of *MacTech Magazine* into your own hands!

For Macintosh Programmers & Developers MacTech™ MAGAZINE

by Nicholas C. "nick.c" DeMello

The NeXT Big Thing

Questions about the NeXT/Apple merger, and where to find the answers — Online.

WHY? — THE TECHNOLOGIES NEXT IS BRINGING TO THE MAC

So, why did Apple choose NeXT? It's no secret that Apple was shopping for new OS technologies. At a minimum, Apple wanted to incorporate preemptive multitasking, protected memory, and symmetric multiprocessing into their new operating system — Rhapsody. All of these features exist in the Mach microkernel (developed at Carnegie Mellon University and then refined by NeXT) and the NeXT OPENSTEP API built on it. Apple's plans for Rhapsody involve a complete implementation of the current MacOS runtime environment (called the Blue Box), as well as a parallel implementation of an OPENSTEP based API (the Yellow Box), both built upon the Mach microkernel. This architecture is outlined on Apple's Rhapsody web site and will allow for compatibility with current software, without relying on emulation. It will also allow the Yellow Box to develop to it's fullest, without being held back by having to maintain compatibility with old system technologies.

However, NeXT brought more than the minimum to the table. Beyond the Mach kernel, and the OPENSTEP based API for interacting with it, NeXT also brought along WebObjects. WebObjects are prebuilt application modules with development tools for combining those objects into custom applications to manage dynamic web based applications. Beyond simple HTML forms, WebObjects allows programmers to develop powerful database frontends and applications. The overview of WebObjects on NeXT's website reminds us that the first world wide web client and server was created using the NeXT technology which evolved into WebObjects — the technology is not only powerful, it's proven with over 10 years of development.

Rhapsody, a Breakdown of the Components of the New OS
<http://www.macos.apple.com/macos/releases/rhapsody/archdiagram.html>
Rhapsody, PDF White Paper
<http://www.apple.be/Acrobat/Rhapsody.pdf>
WebObjects Overview
<http://www.next.com/WebObjects/Overview.html>

HOW? — TOOLS FOR THE NEXT GENERATION

Rhapsody's native language will be Objective-C. In the online MacOS and NeXT technologies FAQ, Apple assures us that they intend to allow developers to create applications for Rhapsody in Java, C, C++, and Pascal — but also mentions that there *will* be advantages to developing in Objective-C.

Those of us not familiar with Objective-C will want to check out the Objective-C world wide web pages, maintained by Steve Dekorte. These pages provide an overview of Objective-C, comments on it's history, listings of reference information, language comparisons, and links to other Objective-C pages. Also, NeXT hosts a series of web pages discussing OOP programming with Objective-C which cover the Objective-C language, extensions, run-time, and an overview of OOP principles. This site also supplies a reference manual and an Objective-C summary.

However, knowing Objective-C won't do us much good without compilers. Metrowerks has accepted that challenge, announcing that they expect to develop an Objective-C compiler for CodeWarrior and develop Objective-C runtime support in their C++ compilers by the 1997 Apple World Wide Developers Conference. According to the Metrowerks web site, they also expect to be able to port CodeWarrior compilers and linkers to Rhapsody's Yellow Box simultaneous with the release of the new OS, getting a developers release of the new tools to developers at MacWorld San Francisco 1998.

MacOS and NeXT Technologies FAQ
<http://macos.apple.com/macos/releases/rhapsody/faq.rhap.html>
The Objective C World Wide Web Pages
<http://www.batech.com/~dekorte/Objective-C/objc.html>
Object Oriented Programming and the Objective C Language
<hhttp://www.next.com/Pubs/Documents/OPENSTEP/ObjectiveC/objctoc.htm>
C Net Article, Metrowerks to Build Rhapsody Tools
<http://www.news.com/News/Item/0,4,6444,00.html>
Metrowerks to Include Rhapsody Tools in Regular CW Subscriptions
<http://www.metrowerks.com/news/press/newos.html>

MT

Nick DeMello is MacTech's Online Presence — you're likely to find him gossiping in usenet space, cruising between web sites, or delving into unexplored ftp vaults in search of new code, interesting resources, and other buried treasures. In this column, Nick shares his most recent internet travels and discoveries with the MacTech audience. Readers are encouraged to return the favor by writing Nick at <URLs@mactech.com> to report online resources for Macintosh programmers. The comprehensive MacTech URL database at <http://www.mactech.com/magazine/urls/> includes, but is not limited to, the URL's reported in MacTech Online.

The Law Office of Bradley M. Sniderman

California Lawyer focusing on Intellectual Property, Corporate, Commercial and Contract law, as well as Wills and Trusts.

If you are looking to protect your software with Copyright or Trademark protection, or if you need help establishing or maintaining your business, please give me a call or an e-mail. Reasonable fees.

(310) 553-4054
Brad@Sniderman.com

MacTech Magazine is your recruitment vehicle

When you need to fill important positions at your company, MacTech Magazine is the consistent choice of companies across the country for hiring the best qualified Macintosh programmers and developers. Let MacTech Magazine deliver your recruit-ment message to an audience of over 27,000 qualified computer professionals.

Call Ruth Subrin at
805/494-9797

"Our Server is loaded."

–MacXperts Programmer & Volleyballer raves about having the best development toys anywhere.

Programmers & Adventurers Wanted

MacXperts is a software developer with immediate openings for Macintosh & Newton programmers with C, C++, OpenDoc, Java or other programming experience.

Take the first step to adventure.
Call Michael Ruttle at:
1-800-356-8040 or fax / 804-358-3847

http://www.macxperts.com
email: work@macxperts.com

NEED: A FEW GOOD MAC DEVELOPERS

Perceptive Scientific Instruments, Inc., based near Houston, Texas, is looking for Mac developers with 2 - 5 years experience to develop its Genetics product line. Strong C/C++ programming and analytical skills are essential. Experience with OOP & digital image processing will be a plus. Send resume to H. Baxi, PSII, 2525 S. Shore Blvd. League City, TX 77573 or fax to (281) 538-2222 or email to hbaxi@persci.com.

MacRegistry™
Developer Job Opportunities

If you are a Macintosh developer, you should register with us! We have a database that enables us to let you know about job opportunities. When we are asked to do a search by a client company the database is the first place we go. There is no charge for registering. The database service is free. Geographic Coverage is nationwide.

Marketability Assessment - To get a specific feel for your marketability send a resumé via Email or call. You may also request a Resume Workbook & Career Planner.

Discreet - We are very careful to protect the confidentiality of a currently employed developer.

Scientific Placement is managed by graduate engineers, we enjoy a reputation for competent & professional job placement services and we are Mac fanatics.

1-800-231-5920 • das@spi.com • Fax 1-800-757-9003

http://www.scientific.com

Scientific Placement, Inc.

MT, Box 19949, Houston, TX 77224, 281-496-6100 Fax: 281-496-0373
MT, Box 71, San Ramon, CA 94583 510-733-6168 beth@spica.bdt.com
MT, Box 202676, Austin, TX 78720-2676 512-260-0123 lej@zilker.net;
Compuserve: 71250,3001; AOL: davesmall

THE TRATTNER NETWORK

The Trattner Network, *"The Digital Talent Source"*, is looking for experienced Macintosh developers for cutting edge opportunities in Northern California and Nationwide.

TTN is plugged in to projects in UI, Internet, Multimedia, Cross platform The hottest development.

NOW — you can visit our web site at: **www.tratnet.com** and see what openings are available for...

- **JAVA/Internet Stars**
- **QA/QC Professionals**
- **Multimedia Developers**
- **Software Developers**
- **Project Coordinator/Manager**
- **Network Professionals**

The Trattner Network has a unique history in **Mac consulting** coupled with exposure to emerging technologies. If you are looking for a chance to enhance your skills and marketability, **please email, fax, or link your resume to:**

The Trattner Network
Attn: Debbie Stevens
170 State St., Suite 240 • Los Altos, CA 94022
Phone (415) 949-9555 ext. 115 - Fax (415) 949-1026
AppleLink: trat.net
email: dstevens@tratnet.com

By Bob Boonstra, Westford, MA

EQUATION EVALUATOR

Those of you with PowerPCs have probably experimented with the Graphing Calculator application installed by default into the Apple Menu Items folder. As one of the first native applications for the PowerPC, the ability of the Graphing Calculator to display, and even animate 2-dimensional and 3-dimensional equations demonstrating the computing power of the PowerPC chip using native code. Underlying the display capability is code to parse an equation and rapidly compute the equation value for a range of input values. In this month's Challenge, you will produce code to perform these parsing and computation functions.

The prototype for the code you should write is:

```
typedef unsigned long ulong;

typedef struct Values {
   float first;        /* first equation input value */
   float delta;        /* first+delta is second input value */
   ulong howMany;      /* number of input values */
} Values;

typedef struct IntValues {
   long first;         /* first equation input value */
   long delta;         /* first+delta is second input value */
   ulong howMany;      /* number of input values */
} IntValues;

typedef struct Results {
   float equationResult;  /* result of equation given x,y,n */
   float x;       /* input value of x producing equationResult */
   float y;       /* input value of x producing equationResult */
   long n;        /* input value of x producing equationResult */
} Results;

void Evaluate(
   char *equation,         /* null-terminated equation to evaluate */
   const Values *xP,       /* equation input values for x */
   const Values *yP,       /* equation input values for y */
   const IntValues *nP,    /* equation input values for n */
   Results w[]             /* preallocated storage for equation values */
);
```

The input equation you are to evaluate will be provided as a null-terminated string in the 2-dimensional curve form (y=x+2x^2) or the 3-dimensional surface form (z=r cos(t) r sin(t)). You are to evaluate the equation for each of the values of x and n (in the case of a 2-dimensional equation) or x, y, and n (in the 3-dimensional case) provided and return the results in the preallocated array w. Each input is described as an initial value, a delta between each value and the next value, and the number howMany of values this input parameter is to assume. For example, if x is to take on the range of values [1.0, 1.1, 1.2, ... 2.0], then the x input could be described as:

```
xP->first = 1.0; xP->delta = .1; xP->howMany = 11
```

In the event that the equation does not contain one of the input parameters, that parameter should be ignored. There is no guarantee, for example, that nP->howMany will be zero when the input equation is not a function of n. Similarly, for a 2-dimensional equation, yP should be ignored.

The input equation might be written as a function of r and θ, which should be calculated from x and y just as the Graphing Calculator does. Because the Graphing Calculator displays equations in ways that cannot be directly represented in a character string, the following symbols will be used in the equation to represent the operator or value indicated:

\	square root
/	division
^	exponentiation
p	pi (π)
t	theta (θ)
.	multiplication (also represented by juxtaposition)

Standard algebraic operator precedence should be used: exponentiation and square roots, then multiplication and division, then addition and subtraction, with left-to-right evaluation order for operators of equal precedence, and with parentheses used to override normal precedence. Arguments to trigonometric functions will always be surrounded by parentheses. Where the Graphing Calculator would use superscripts to indicate an extended exponent, parentheses will be used to make the meaning clear (e.g., e^(x+y)).

Store the equation result for the i-th combination of input values in w[i]->equationResult, and store the corresponding input values in w[i]->x, w[i]->y, and w[i]->n. The results may be stored in any order, but each input combination should appear exactly once. Results should be calculated with a minimum relative accuracy of .00001.

Even though the Graphing Calculator handles inequalities, multiple equations, differentiation, simplification, and expansion, your code does not need to deal with these cases. With these exceptions, your code should handle any equation that the Graphing Calculator can handle.

You may allocate any amount of dynamic storage up to 20MB, provided you deallocate the storage before returning. This will be a native PowerPC Challenge, using the CodeWarrior environment. Solutions may be coded in C, C++, or Pascal. Limited use of assembly language is also permitted, for anyone who might need it to access any dynamically generated code as part of their solution. The solution computing the correct results in the least amount of time will be the winner.

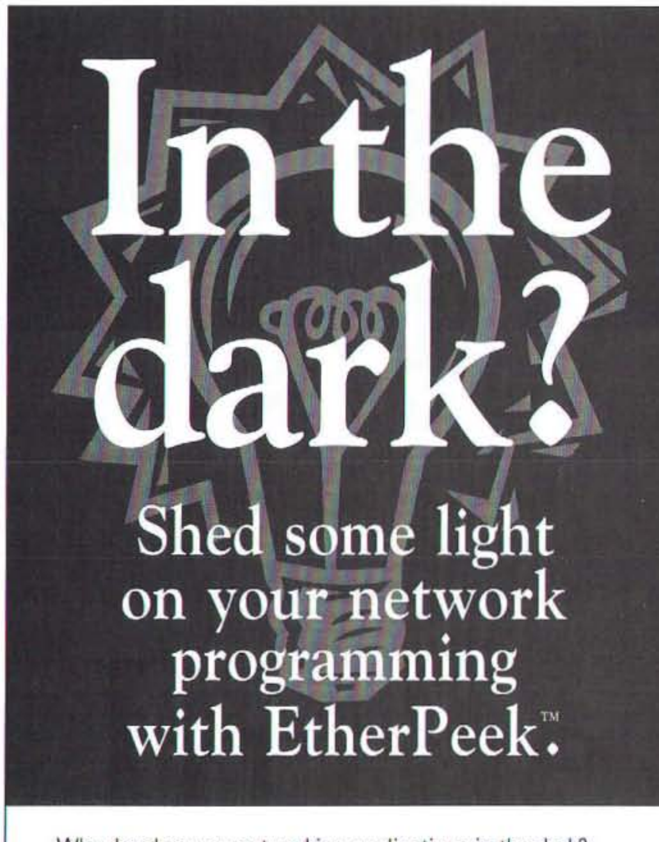
Three Months Ago Winner

Congratulations to **Jeff Mallett** (Boulder Creek, CA) for producing the winning entry among ten submitted for the Othello Challenge. The objective was to win a round-robin Othello tournament, generalized to allow a board size between 8 and 64 (even numbers only), by as large a margin as possible, using as little computation time as possible. Tournament points were based on the margin of victory (the number of the winner's pieces showing at the end of the game minus the number of the opponent's pieces showing), and on the amount of computation time used, as follows:

[margin of victory - seconds of execution time / 30] / number of squares

The test cases included board sizes of 8x8 and 18x18, with each player competing against each other player twice, once playing first, with the black pieces, and once playing second, with the white pieces. I had planned to run some larger cases, but the first two still had not completed after running all night, so I had to stop at 18x18.

The solutions submitted varied considerably in complexity. The simplest (and fastest) solutions assigned values to each board position and made the move which maximized total value. Some solutions took that approach one step further and evaluated an opponent's potential responses and used a min/max approach to select the best move. Other solutions took credit for the number of pieces flipped on a move, with possible consideration for



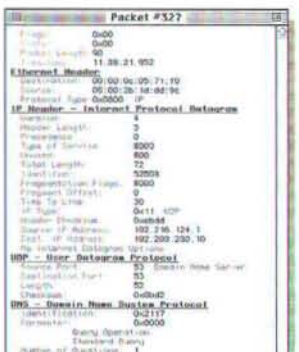
In the dark?

Shed some light on your network programming with EtherPeek.™

Why develop your networking applications in the dark?

Use **EtherPeek Lite**, Ethernet protocol analysis software, to test and debug your software and hardware communications in minutes instead of hours!

- Check for protocol compliance.
- Use hundreds of built-in decodes. (IP, IPv6, IGRP, Netware IPX/SPX, NetBEUI, NetBIOS, DECnet, AppleTalk)
- Develop custom packet decoders.
- Filter packets during or after capture.
- Test device reactions to specific packets.
- Customize or alter packets for transmission.
- Generate traffic to test varying loads.
- Write your own plug-in modules.



```
Packet #327
  [illegible]             0x00
  [illegible]             0x00
  Packet Length:          90
  Timestamp:              11:38:21.952
Ethernet Header
  Destination:            00:00:0c:05:71:19
  Source:                 06:00:3b:1d:dd:9c
  Protocol Type:          0x0800  IP
IP Header - Internet Protocol Datagram
  Version:                      4
  Header Length:                5
  Precedence:                   0
  Type of Service:              %000
  Unused:                       %00
  Total Length:                 72
  Identifier:                   52509
  Fragmentation Flags:          %000
  Fragment Offset:              0
  Time To Live:                 30
  IP Type:                      0x11  UDP
  Header Checksum:              0xebdd
  Source IP Address:            192.216.124.1
  Dest. IP Address:             192.203.230.10
  No Internet Datagram Options
UDP - User Datagram Protocol
  Source Port:                  53  Domain Name Server
  Destination Port:             53
  Length:                       52
  Checksum:                     0x0bd2
DNS - Domain Name System Protocol
  Identification:               0x2117
  Parameter:                    0x0000
                Query Operation
                Standard Query
  Number of Questions:          1
  Number of Answers:            0
  Number of Authority:          0
  Number of Additional:         0
  Query Domain Name:            75.72.161.206.in-addr.arpa
  Query Type:                   12  Domain Name Pointer
  Query Class:                  1  Internet
Frame Check Sequence:  0x26b3bf0e
```

ORDER TODAY!

Mention This Ad For Your Special US $399.00 EtherPeek Lite Price.

CALL: 800/466-AGGP or (1)510/937-7900
FAX: (1)510/937-2479
EMAIL: eplite-info@aggroup.com
WWW: http://www.aggroup.com/eplite/

ag

Features | Benefits

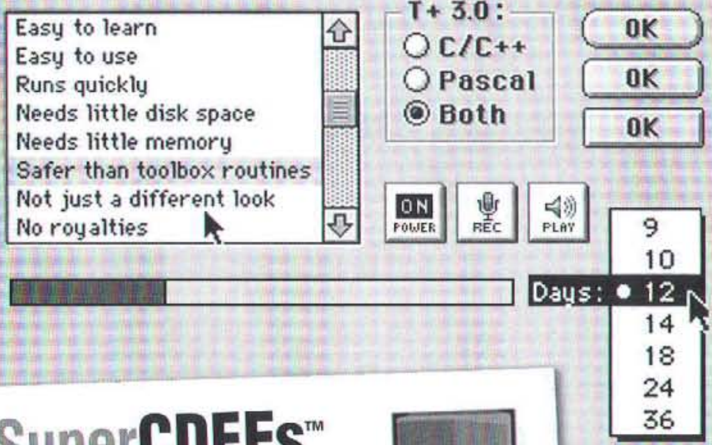
Create a *working* GUI element with just 1 line of code!

Tools Plus 3.1

Get extraordinary results using ordinary C, C++ or Pascal. Only 170 core routines (about 370 total) replace the need for thousands of Toolbox routines, and thousands of lines of code. Tools Plus has you covered with…

- **Windows** (all kinds, auto-positioning)
- **Cursors** (color, animation, auto-change)
- **Scroll bars** (speed control, live scrolling)
- **Picture buttons** (the best anywhere)
- **Makes CDEFs & LDEFs work automatically**
- **Panels** (3D, flat, many options)
- **Menus** (pull-down and hierarchical)
- **Editing fields** (w/scroll bars and filters)
- **Edit menu** (auto-editing with undo/redo)
- **Full color and multi-monitor support**
- **World-class custom buttons and sliders**
- **Tool bar**
- **Floating palettes**
- **Buttons** (all kinds)
- **List boxes**
- **Pop-up menus**
- **Clipboard**
- **Dynamic alerts** (no resources req'd)
- **Automatic event handling**
- **and much more!**

One line creates it. One line makes it work.



SuperCDEFs™

World-Class Controls for the Discerning Developer

For all Macintoshes and system versions

Full color and multi-monitor support

680x0, PowerMac and Fat/Safe format

The finishing touch for a complete 3D look

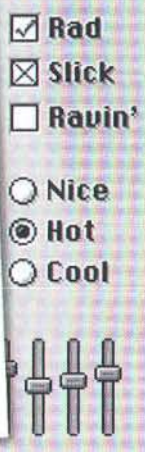
✓ 7 buttons, 9 tabs and 11 sliders (flat/3D text, flat/3D body, raised/inset title, soft/bold shadows, variable shapes) plus a thermometer

✓ Customizable check boxes plus undefined state

✓ Sliders options: tick marks (1 side/2 sides/none), scale (forward/reversed/none), snap or step to mouse, smart scaling

only $89*

***Free with purchase of Tools Plus Developer Kit**



Water's Edge Software

2441 Lakeshore Road West
Box 70022
Oakville, Ontario
Canada L6L 6M9
Phone: (416) 219-5628
Fax: (905) 847-1638

e-mail: WaterEdgSW@aol.com
http://www.interlog.com/~wateredg

Tools Plus for	
Symantec (THINK) C/C++	**$149**
THINK Pascal	**$149**
THINK C/C++ & Pascal	**$199**
CodeWarrier Bronze	**$199**
CodeWarrier Gold	**$249**

(We accept VISA and Amex. Add $10 for shipping.)
*Call for Academic pricing

Free Evaluation Kit: Available on AOL, CIS, Internet and at our web site

MW ★★★★★ **Macworld, Feb'96**

vulnerability to reversal on the next move. The winning solution used the most complicated approach, with lines of play being examined by a progressively deepening search.

The table below describes how each player fared against each other player. Each row shows the number of victories that player had against the player represented by the corresponding column. The final column shows the total number of victories won out of the 36 games played by each entry. As you can see, the second place entry by **Randy Boring** won nearly as many games as Jeff's winning entry, and actually beat the winning entry in one of the four games they played.

Player	Name	1	2	3	4	5	6	7	8	9	10	Wins
1	David Whitney	–	1	4	2	4	2	2	0	0	4	19
2	Dan Farmer	3	–	4	2	4	4	4	1	0	4	26
3	David McGavran	0	0	–	2	4	1	1	0	1	1	10
4	Eric Hangstefer	2	2	2	–	4	0	1	0	0	3	14
5	Ken Slezak	0	0	0	0	–	0	0	0	0	0	0
6	Gregory Cooper	2	0	3	4	4	–	2	0	1	4	20
7	Mason Thomas	2	0	3	3	4	2	–	0	0	3	17
8	Randy Boring	4	3	4	4	4	4	4	–	1	4	32
9	Jeff Mallett	4	4	3	4	4	3	4	3	–	4	33
10	Ludovic Nicolle	0	0	3	1	4	0	1	0	0	–	9

The top two entries used significantly more computation time than the others. Jeff's winning entry used an average of more than one second per move (as you can see by examining the parameter settings in his code), but the larger margin of victory more than offset the execution time penalty.

The table below provides the code and data sizes for each of the solutions submitted, along with the total number of victories won in all of the test cases, and the cumulative score earned in those victories. Numbers in parenthesis after a person's name indicate that person's cumulative point total for all previous Challenges, not including this one.

Player	Code	Data	Wins	Score
Jeff Mallett (44)	6988	277	33	25.49
Randy Boring (27)	6908	64	32	20.37
Gregory Cooper (27)	4764	284	20	15.00
Dan Farmer	9240	96	26	14.27
David Whitney	7216	864	19	9.79
Eric Hangstefer	4124	80	14	9.72
Mason Thomas (4)	6976	57	17	9.01
David McGavran	3272	48	10	8.09
Ludovic Nicolle (21)	6436	120	9	6.33
Ken Slezak (20)	9256	77	0	0.00

SAMPLE GAME

Here is one of the games played by the top two entries. Randy Boring's entry played Black, and Jeff Mallett's played White. The moves are given as the row and column selected by the programs, interspersed with an occasional view of the resulting board position.

Move	Black (Boring)		White (Mallett)	
	row	col	row	col
1	2	3	4	2
2	5	2	2	4
3	2	5	3	5
4	4	5	3	6
5	2	2	3	2
6	2	6	5	3

```
- - - - - - - -
- - - - - - - -
- - B B B B B -
- - W W W B W -
- - W W B B - -
- - B W - - - -
- - - - - - - -
- - - - - - - -
```

7	3	1		

… missing an opportunity to place a piece on the edge at (2,7), inviting (1,7) as a response by white, in hope of moving to (0,7)

			5	4
8	4	1	1	5
9	3	7		

```
- - - - - - - -
- - - - - W - -
- - B B B W B -
- B B B B B B B
- B B B B W - -
- - B W W - - -
- - - - - - - -
- - - - - - - -
```

Black occupies the first edge square.

			2	7
10	1	7	1	3
11	1	2	4	6
12	0	3	5	5
13	6	2	0	2

```
- - W B - - - -
- - B W - W - B
- - B B W W B B
- B B W B B W B
- B B W B W W -
- - B B W W - -
- - B - - - - -
- - - - - - - -
```

Black's next move, to (0,1), prevents White from obtaining a foothold on one of the edges.

14	0	1	2	1
15	3	0	2	0
16	1	0	5	0
17	5	1	6	3
18	0	4	4	0
19	6	0	7	2
20	1	4	1	6

```
- B B B B - - -
B - B B B B W B
B W B B B W W B
B B B W B B W B
B B W W B W W -
B B W W W W - -
B - W W - - - -
- - W - - - - -
```

While white had other choices, none of them were very good, and this move to (1,6) gives Black a corner on the next move.

21	0	7	6	1
22	7	0		

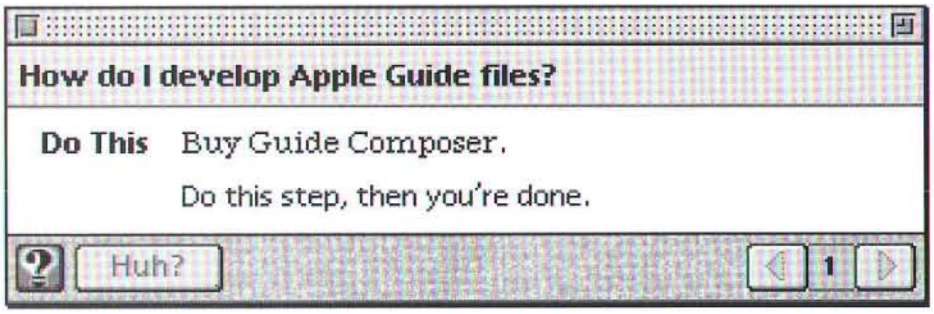
Black takes a second corner …

```
- B B B B - - B
B - B B B B B B
B W B B B B W B
B W B W B B W B
B W W B B W W -
B W B W W W - -
B B W W - - - -
B - W - - - - -
```

			7	1
23	7	3	7	4
24	6	5	6	4
25	7	5	6	6
26	7	7		

… and a third …

```
- B B B B - - B
B - B B B B B B
B W B B B B W B
B B B W B B W B
B W B B B B W -
B B W B B B - -
B W B W W W B -
B B B B B B - B
```


How do I develop Apple Guide files?

Do This Buy Guide Composer.

Do this step, then you're done.

Available from the MacTech Mail Order Store

Demo/info:http:rampages.onramp.net/~stepup

$99

STEPUP SOFTWARE
7110 Glendora Avenue
Dallas, Texas 75230
214-360-9301
214-360-0127 fax

Guide Composer is a trademark of StepUp Software. Apple Guide is a trademark of Apple Computer, Inc.

```
                          5     6
27        7     6         6     7
28        1     1

      -  B  B  B  B  -  -  B
      B  B  B  B  B  B  B  B
      B  B  B  B  B  B  W  B
      B  B  B  W  B  B  W  B
      B  W  B  B  B  B  W  -
      B  B  W  W  B  W  W  -
      B  W  B  W  W  W  W  W
      B  B  B  B  B  B  B  B
```

Black could have done better with (5,7) at this point, instead of giving the final corner to White.

```
                                00
        29        4       7     57
        30        –       –     05
        31        0       6

      W  W  W  W  W  W  B  B
      B  W  B  B  W  B  B  B
      B  B  W  W  B  W  W  B
      B  B  W  W  B  W  B  B
      B  W  B  B  B  W  B  B
      B  B  W  W  B  W  W  W
      B  W  B  W  W  B  W  W
      B  B  B  B  B  B  B  B
```

Black wins by a score of 37 to 27, a relatively close game compared to many in the tournament.

TOP 20 CONTESTANTS

Here are the point totals for the Top Contestants in the Programmer's Challenge. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

Rank	Name	Points	Rank	Name	Points
1.	Munter, Ernst	184	11.	Cutts, Kevin	21
2.	Gregg, Xan	114	12.	Nicolle, Ludovic	21
3.	Larsson, Gustav	47	13.	Picao, Miguel Cruz	21
4.	Lengyel, Eric	40	14.	Brown, Jorg	20
5.	Boring, Randy	37	15.	Gundrum, Eric	20
6.	Cooper, Greg	34	16.	Higgins, Charles	20
7.	Lewis, Peter	32	17.	Kasparian, Raffi	20
8.	Mallett, Jeff	27	18.	Slezak, Ken	20
9.	Antoniewicz, Andy	24	19.	Studer, Thomas	20
10.	Beith, Gary	24	20.	Karsh, Bill	19

There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place........20 points	5th place ....2 points
2nd place ......10 points	finding bug ....2 points
3rd place ....7 points	suggesting Challenge...2 points
4th place........4 points	

StoneTable Break on through the limitations of the List Manager

	Name	Home Address
Father	**John Smith**	3224 NE 58th Avenue P.O Box 12665 Portland, OR 97212

Row and column titles
variable size row and columns
move, copy, sort, hide, resize rows and columns
set font, size, fore/back color, face, alignment per cell
cell margins, top/bottom & left/right
drag cells in and between tables
edit cells in place
popup menus & check boxes
draw boxes around multiple cells
variable size grid lines
PICT's and controls in cells
"LDEF-like" custom drawing function
greater than 32K data per table
plus all List Manager functions and more

68 & PPC libraries for Think C, CodeWarrior C & Pascal, MPW C & Pascal $200.00

Includes PowerPlant Class for CodeWarrior
International shipping (US Airmail) $10
No royalty fees for applications

StoneTablet Publishing
P.O. Box 12665
Portland, OR 97212
voice/fax (503) 287-3424
stack@teleport.com

New Price
New Version

More Info - http://www.teleport.com/~stack
Demo - ftp://ftp.teleport.com/pub/vendors/stack/StoneTableDemo.hqx

---

Here is Jeff's winning solution:

```
*Jello.c An nxn Othello program in C
* Copyright 1997 Jeff Mallett
/*
 * Uses alpha-beta search with iterative deepening, transposition tables,
 * extension for solving, futility cut-off, a simplistic selectivity, etc.
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MY_ASSERT(b)
```

Engine Parameters

```
  // Extensions/Pruning
  #define kFullDepthPlies       2  // # of full-depth plies before selectivity
  #define kSolveThreshold       4  // Plies extension for solving whole board
  #define kMinimumSafeDisks  3  // Minimum disks to have and still be safe
  #define kFutilityScore     kCorner
                              // Score above beta to trigger futility cut-off
  #define kWipeOutExtension   2
             // Extend a ply if opponent doesn't have more disks than this

  // Average move time in 60ths of a second
  #define kOpeningMoveTime   50          // In the opening
  #define kMoveTime                150   // Normally
  #define kSolveMoveTime     800         // When trying to solve

  // Scoring Parameters
  #define kFinalDisk     50              // per disk on board at the end
  #define kTooFewDisks   -50             // per disk under threshold
```

Constants/Macros

```
#define kInfinity     110000000L
#define kBestPossible 100000000L

// Directions (if se increases both x and y):
//7 6 5 4 3 2 1 0
// NE SW NW SE N S W E (opposites are adjacent)
enum { DIR_E = 0, DIR_W, DIR_S, DIR_N,
           DIR_SE, DIR_NW, DIR_SW, DIR_NE };

enum { E  = 0x0001,
       W  = 0x0002,
       S  = 0x0004,
       N  = 0x0008,
       SE = 0x0010,
       NW = 0x0020,
       SW = 0x0040,
       NE = 0x0080,

       E_BORDER  = 0x0100,
       W_BORDER  = 0x0200,
       S_BORDER  = 0x0400,
       N_BORDER  = 0x0800,
       SE_BORDER = 0x1000,
       NW_BORDER = 0x2000,
       SW_BORDER = 0x4000,
       NE_BORDER = 0x8000
};

// Array of squares on the board (up to 66 x 66 squares)
// Each square has:
// 1st 8 bits — In this direction there's a BLACK disk that can be flipped by WHITE
#define ADJACENCY           0x000000FF // Adjacent to disk in 8 directions
#define BORDER_ADJACENCY 0x0000FF00 // Adjacent to border in 8 directions
#define WHITE               0x00010000 // Is White disk here?
#define BLACK               0x00020000 // Is Black disk here?
#define COLOR_BITS          0x00030000 // Mask: Is disk here?
#define BORDER_BIT          0x00040000 // Is this border square?
#define COLOR_BORDER_BITS   0x00070000 // Mask: Is border or disk here?
#define BAD_BIT             0x00080000 // X or C square
#define EMPTYADJ_BITS       0xFFF00000    /* Top 12 bits (0-4095) */

#define BORDER_ADJACENCY_SHIFT 8
#define COLOR_SHIFT            16
#define EMPTYADJ_SHIFT         20

#define IS_NOT_EMPTY(z)     ((z) & COLOR_BORDER_BITS)
#define IS_EMPTY(z)         !IS_NOT_EMPTY(z)
#define HAS_DISK(z)         ((z) & COLOR_BITS)
#define HAS_NO_DISK(z)      !HAS_DISK(z)
#define IS_BORDER(z)        ((z) & BORDER_BIT)
#define IS_ON_BOARD(z)      !IS_BORDER(z)
#define IS_BAD(z)           ((z) & BAD_BIT)
#define IS_EDGE(z)          ((z) & BORDER_ADJACENCY)
#define IS_CORNER(z)    (gCountZeros[((z) & BORDER_ADJACENCY) \
                        >> BORDER_ADJACENCY_SHIFT] == 3)
```

```
#define OPPONENT(side)      ((side) ^ COLOR_BITS)

#define XY2INDEX(x, y)      ((y) * gRealBoardSize + (x))

#define COUNT_EMPTIES(z) \
  gCountZeros[ ((z) | ((z) >> \
                        BORDER_ADJACENCY_SHIFT)) & ADJACENCY ]

#define DIR_BIT(dir)          (1L << (dir))
#define OPP_DIR_BIT(dir)   gOppDirBit[dir]

#define EMPTYADJ_BIT(index)  \
                  ((index) << EMPTYADJ_SHIFT)
#define GET_EMPTYADJ_INDEX(pSq)\
                  (*(pSq) >> EMPTYADJ_SHIFT)
#define SET_EMPTYADJ_INDEX(pSq, index) \
  *(pSq) = (*(pSq) & ~EMPTYADJ_BITS) | EMPTYADJ_BIT(index)

// Add to end of list
#define ADD_TO_EMPTYADJ(pSq) \
  SET_EMPTYADJ_INDEX(pSq, gSizeEmptyAdj); \
  gEmptyAdj[gSizeEmptyAdj++] = pSq

// Swap entry in list with end entry and shrink list
#define REMOVE_FROM_EMPTYADJ(pSq) { \
  long i = GET_EMPTYADJ_INDEX(pSq); \
  unsigned long *p = gEmptyAdj[--gSizeEmptyAdj]; \
  gEmptyAdj[i] = p; \
  SET_EMPTYADJ_INDEX(p, i); \
}

#define PUSH(x)         *(gChangesEnd++) = (x)
#define START_SAVE      PUSH(0L)
#define PUSH_SQ(pSq) \
        { PUSH(*(pSq)); PUSH((unsigned long)(pSq)); }
#define POP             *(--gChangesEnd)
#define TOP             *gChangesEnd

typedef unsigned long * PSQUARE;
typedef long SCORE;

#define USE_HASH
#ifdef USE_HASH
  #define kHashTableMask   0x7FFF // 32K entry table
  #define kHashTableSize   (kHashTableMask + 1)
  #define kSwitchSideHash     0x87654321
  enum { INVALID = 0, VALID = 1,
            LOWER_BOUND = 2, UPPER_BOUND = 3 };

  typedef struct SHash {
    unsigned long HashValue;
    SCORE Score;
    PSQUARE BestMove;
    short Depth;
    short Type;
  } SHash;

  static SHash *gHashTable;
  static long gWhiteHashOffset, gBlackHashOffset,
              gFlipHashOffset;
  static unsigned long gHashValue;
#endif
```

Prototypes

```
Boolean /*legalMove*/ Othello (
  long boardSize,       /* number of rows/columns in the game board */
  long oppRow,          /* row where opponent last moved, 0 .. boardSize-1 */
  long oppCol,          /* column where opponent last moved, 0 .. boardSize-1 */
  long *moveRow,        /* return your move - row, 0 .. boardSize-1 */
  long *moveCol,        /* return your move - column, 0 .. boardSize-1 */
  void *privStorage,  /* preallocated storage for your use */
  long storageSize,   /* size of privStorage in bytes */
  Boolean newGame,      /* TRUE if this is your first move in a new game */
  Boolean playBlack     /* TRUE if you play first (black pieces) */
);

static SCORE Search(SCORE alpha, SCORE beta, unsigned long
side, long depth, Boolean passEndsGame);
static long Generate(unsigned long side);
static SCORE MakeMove(PSQUARE to, unsigned long side);
static void UnmakeMove();
static void Initialize(long boardSize, void *privStorage);
static SCORE Evaluate(unsigned long side);
static long SortValue(PSQUARE p, unsigned long side);
static void BubbleSort(long n, unsigned long side);
```

Static Variables

```
// Direction indices
static unsigned long gOppDirBit[8] =
       { W, E, N, S, NW, SE, NE, SW };

// Offsets on board plus zero sentinel
// Fill in gOffsets[2..7] when we know board size
static long gOffsets[9] = {1L,-1L,99L,99L,99L,99L,99L,99L,0L};

// gSquares — Array of squares: Stores board
static unsigned long *gSquares;
static unsigned long *gOnBoardStart;  // Pointer into gSquares
static unsigned long *gOnBoardEnd;    // Pointer into gSquares
static long gNumOnSquares;   // # of squares, not including borders
static long gRealBoardSize;  // Length of a side (includes borders)

// gEmptyAdj — Array of pointers to squares:
//   Stores all empty squares adjacent to disk(s)
static PSQUARE *gEmptyAdj;
static long gSizeEmptyAdj;

// gChanges — Array of unsigned longs containing data to undo moves
//   list of:
//       <pointer to square> <old square value>
//   terminated by a 0L
// The first position will be the drop square and the others will be flips
static unsigned long *gChanges;
static unsigned long *gChangesEnd; // Pointer into gChanges

// gCountZeros — Precalculated 256-element constant array
//   returns count of zero bits in the byte
static unsigned long *gCountZeros;

// gTree — Array of pointers to squares: Holds search tree
static PSQUARE *gTree;
static PSQUARE *gTreeEnd; // Pointer into gTree

// gMobility — Array of counts of moves available at each ply
static long *gMobility;

// Pointers to various special squares
static PSQUARE gNWCorner, gNWX, gNWC1, gNWC2;
static PSQUARE gNECorner, gNEX, gNEC1, gNEC2;
static PSQUARE gSWCorner, gSWX, gSWC1, gSWC2;
static PSQUARE gSECorner, gSEX, gSEC1, gSEC2;
//The gCounts array holds current counts of disks
//Access is by gCounts[side >> COLOR_SHIFT]
// gCounts[WHITE_INDEX] white's disks
// gCounts[BLACK_INDEX] black's disks
#define WHITE_INDEX     (WHITE >> COLOR_SHIFT)
#define BLACK_INDEX     (BLACK >> COLOR_SHIFT)
static unsigned long gCounts[3];

static SCORE gIncScore;             // Score relative to side to move
static SCORE gEndgameDiskBonus;  // Bonus per disk in endgame
static SCORE kX, kC, kEdge, kCorner; // Penalties/Bonuses

static long gAbortSearchTime;   //Time at which the search will be aborted
static Boolean gAborted;        // Has the search been aborted?

static long gStartDepth;        // Search was started at this depth
static long gPly;               // Number of moves deep
```

Othello

```
Boolean /*legalMove*/ Othello (
  long boardSize,      /* number of rows/columns in the game board */
  long oppRow,         /* row where opponent last moved, 0 .. boardSize-1 */
  long oppCol,       /* column where opponent last moved, 0 .. boardSize-1 */
  long *moveRow,       /* return your move - row, 0 .. boardSize-1 */
  long *moveCol,       /* return your move - column, 0 .. boardSize-1 */
  void *privStorage,  /* preallocated storage for your use */
  long storageSize,   /* size of privStorage in bytes */
  Boolean newGame,     /* TRUE if this is your first move in a new game */
  Boolean playBlack    /* TRUE if you play first (black pieces) */
)
{
  PSQUARE *to, p;
  unsigned long side = playBlack ? BLACK : WHITE;
  unsigned long nextSide;
  SCORE t, bestScore, saveIncScore;
  long j, generated, index, x, y;
  long stillOpen, bestFoundAt, *pOffsets;
  long startTime, targetTime, targetDuration;
  Boolean nearEdge;
#ifdef USE_HASH
  unsigned long saveHashValue;
#endif

  if (newGame) {
    Initialize(boardSize, privStorage);
  }
  gChangesEnd = gChanges;

#ifdef USE_HASH
  // Fix up gHashTable
  {
    SHash *pHashTable = gHashTable;
    int i;

    i = kHashTableSize - 1;
    do {
      (pHashTable++)->Depth -= 2;
    } while (i--);
  }
#endif

  if (oppRow != -1) {
    index = XY2INDEX(oppCol+1, oppRow+1);
    gIncScore -=
      MakeMove(&gSquares[index], playBlack ? WHITE : BLACK);
    gChangesEnd = gChanges;
  }

  gTreeEnd = gTree;
  generated = Generate(side);

  if (!generated) { // No moves
    *moveRow = *moveCol = -1;
    return TRUE;
  }
  if (generated > 1 && !(newGame && oppRow == -1)) {
    BubbleSort(generated, side);

    gMobility[0] = gMobility[1] = generated;
    gPly = 1;

    nextSide = OPPONENT(side);
    stillOpen = gNumOnSquares - gCounts[WHITE_INDEX] -
                                  gCounts[BLACK_INDEX];

    // Calculate gEndgameDiskBonus
    gEndgameDiskBonus = 0;
    x = gNumOnSquares / 3;
    j = x - stillOpen;
    if (j > 0) {
      gEndgameDiskBonus = (j * kFinalDisk) / (x * 5);
      if (!gEndgameDiskBonus)
        gEndgameDiskBonus = 1;
    }

    // Do we have any pieces near an edge?
    nearEdge = false;
    for (p = gOnBoardStart;
        !nearEdge && p <= gOnBoardEnd; ++p) {
      if (HAS_DISK(*p)) {
        if (IS_EDGE(*p)) {
          nearEdge = true;
        } else {
          pOffsets = gOffsets;
          do {
            if (IS_EDGE(*(p + *pOffsets)))
              nearEdge = true;
          } while (!nearEdge && *(++pOffsets));
        }
      }
    }

    // Set times
    if (!nearEdge)
      targetDuration = kOpeningMoveTime;
    else if (stillOpen > 20)
      targetDuration = kMoveTime;
    else // try to solve!
      targetDuration = kSolveMoveTime;
    startTime = LMGetTicks();
    targetTime = startTime + targetDuration;
    gAbortSearchTime = startTime + targetDuration * 6;
    gAborted = false;

    saveIncScore = gIncScore;
#ifdef USE_HASH
    saveHashValue = gHashValue;
#endif

    for (gStartDepth=1; gStartDepth < stillOpen && !gAborted;
                                                    ++gStartDepth) {
      to = gTree;
      bestScore = -kInfinity;
      for (j=0; j<generated; ++j) {
        gIncScore = - (gIncScore + MakeMove(*to, side));
          ++gPly;
          t = -Search(-kInfinity, kInfinity, nextSide,
                          gStartDepth - 1, false);
          --gPly;
        UnmakeMove();
        gIncScore = saveIncScore;
#ifdef USE_HASH
        gHashValue = saveHashValue;
#endif
        if (gAborted)
          break;

        if (t > bestScore) {
          PSQUARE bestMove, *p;
          // Move *to to front of the tree
          bestMove = *to;
          MY_ASSERT(bestMove >= gOnBoardStart && bestMove <=
                                          gOnBoardEnd);

          for (p = to; p != gTree; --p)
            *p = *(p-1);
          *gTree = bestMove;

          bestScore = t;
          bestFoundAt = gStartDepth;
        }
        if (LMGetTicks() >= targetTime) {
          if (bestScore > -kInfinity && gStartDepth +
                    kSolveThreshold + 1 != stillOpen)
            break; // time to stop
          if (LMGetTicks() - startTime
            >= 3 * targetDuration)
            break; // time to stop
        }
        ++to;
      }

      if (gStartDepth >= 4 && LMGetTicks() - startTime > 1 +
          targetDuration/2)
        break; // probably not enough time to finish another iteration
      if (gStartDepth - bestFoundAt == 3
          && IS_CORNER(*gTree[0]))
        break; // easy corner move
    }
  }

  gIncScore += MakeMove(*gTree, side);
  index = (long)(*gTree - gSquares);
  y = index / gRealBoardSize;
  x = index - y * gRealBoardSize;
  *moveCol = x - 1;
  *moveRow = y - 1;

  return TRUE;
}
```

SortValue

```
// Returns value that orders squares for root search
long SortValue(PSQUARE p, unsigned long side)
{
  long stillOpen, value;
  PSQUARE q;
  unsigned long occupant, enemy;
  long *pOffsets;

  if (IS_EDGE(*p)) {
    if (IS_CORNER(*p))
      return 200; // Corner
    if (IS_BAD(*p))
      return -100; // C
    return 100; // Edge
  }
  if (IS_BAD(*p))
    return -200; // X

  // Check p's adjacency bits
  stillOpen = gNumOnSquares - gCounts[WHITE_INDEX] -
                                        gCounts[BLACK_INDEX];
  enemy = OPPONENT(side);
  value = 0;
  pOffsets = gOffsets;
  do {
    q = p + *pOffsets;
    occupant = *q & COLOR_BITS;
    if (stillOpen > 10) {
      if (occupant == side)
        ++value; // good to take away empty-adjacent
      else if (occupant == enemy)
        --value; // bad to flip
    } else if (occupant == OPPONENT(side))
      ++value; // endgame: good to flip
  } while (*(++pOffsets));

  return value;
}
```

BubbleSort

```
// Sorts generated root moves in decreasing value order
// Hey, bubble sort is really okay in this case
void BubbleSort(long n, unsigned long side)
{
  long i, j, swaps;
  PSQUARE temp;

  for (j=n-2; j>=0; --j) {
    swaps = 0;
    i = 0;
    do {
      if (SortValue(gTree[i+1], side)
          > SortValue(gTree[i], side)) {
        ++swaps; // Swap i and i+1
        temp = gTree[i];
        gTree[i] = gTree[i+1];
        gTree[i+1] = temp;
      }
    } while (i++ != j);
    if (!swaps)
      break; // Already sorted: Finish early
  }
}
```

Evaluate

```
// Evaluate position and return score relative to side
// side is also the side to move
SCORE Evaluate(unsigned long side)
{
  SCORE score = 0;

  // Maximize disks, but only in endgame
  if (gEndgameDiskBonus) {
    score = (gCounts[WHITE_INDEX] - gCounts[BLACK_INDEX]) *
gEndgameDiskBonus;
    if (side == BLACK)
      score = -score;
  }

  // NW Corner Area
  if (HAS_DISK(*gNWCorner)) {
    score += (*gNWCorner & side) ? kCorner : -kCorner;
  } else if (*gNWCorner & ADJACENCY) {
    if (HAS_DISK(*gNWX)) {
      score += (*gNWX & side) ? kX : -kX;
    }
    if (HAS_DISK(*gNWC1)) {
      score += (*gNWC1 & side) ? kC : -kC;
    }
    if (HAS_DISK(*gNWC2)) {
      score += (*gNWC2 & side) ? kC : -kC;
    }
  }

  // NE Corner Area
  if (HAS_DISK(*gNECorner)) {
    score += (*gNECorner & side) ? kCorner : -kCorner;
  } else if (*gNECorner & ADJACENCY) {
    if (HAS_DISK(*gNEX)) {
      score += (*gNEX & side) ? kX : -kX;
    }
    if (HAS_DISK(*gNEC1)) {
      score += (*gNEC1 & side) ? kC : -kC;
    }
    if (HAS_DISK(*gNEC2)) {
      score += (*gNEC2 & side) ? kC : -kC;
    }
  }

  // SW Corner Area
  if (HAS_DISK(*gSWCorner)) {
    score += (*gSWCorner & side) ? kCorner : -kCorner;
  } else if (*gSWCorner & ADJACENCY) {
    if (HAS_DISK(*gSWX)) {
      score += (*gSWX & side) ? kX : -kX;
    }
    if (HAS_DISK(*gSWC1)) {
      score += (*gSWC1 & side) ? kC : -kC;
    }
    if (HAS_DISK(*gSWC2)) {
      score += (*gSWC2 & side) ? kC : -kC;
    }
  }

  // SE Corner Area
  if (HAS_DISK(*gSECorner)) {
    score += (*gSECorner & side) ? kCorner : -kCorner;
  } else if (*gSECorner & ADJACENCY) {
    if (HAS_DISK(*gSEX)) {
      score += (*gSEX & side) ? kX : -kX;
    }
    if (HAS_DISK(*gSEC1)) {
      score += (*gSEC1 & side) ? kC : -kC;
    }
    if (HAS_DISK(*gSEC2)) {
      score += (*gSEC2 & side) ? kC : -kC;
    }
  }
    // Too few disks?
  if (gCounts[WHITE_INDEX] < kMinimumSafeDisks) {
    SCORE x = (kMinimumSafeDisks - gCounts[WHITE_INDEX]) *
                                     kTooFewDisks;
    score += (side == WHITE) ? x : -x * 2;
  }
  if (gCounts[BLACK_INDEX] < kMinimumSafeDisks) {
    SCORE x = (kMinimumSafeDisks - gCounts[BLACK_INDEX]) *
                                     kTooFewDisks;
    score += (side == BLACK) ? x : -x * 2;
  }

  // Mobility
  score += gMobility[gPly-2] - gMobility[gPly-1];

  // Could also have a value for the right to move

  return gIncScore + score;
}
```

Search

```
SCORE Search(SCORE alpha, SCORE beta, unsigned long side,
long depth, Boolean passEndsGame)
{
  register PSQUARE *to;
  unsigned long nextSide;
  long generated;
  SCORE t, bestScore, saveIncScore;
  PSQUARE *firstMove;
  long stillOpen;
  SCORE oldAlpha = alpha;
#ifdef USE_HASH
  unsigned long saveHashValue;
  PSQUARE bestMove, tableReply;
  SHash *pHashTable;
#endif

  stillOpen = gNumOnSquares - gCounts[WHITE_INDEX] -
                                   gCounts[BLACK_INDEX];
  if (!stillOpen) {
    // Board full
    bestScore = (gCounts[WHITE_INDEX] -
                       gCounts[BLACK_INDEX]) * kFinalDisk;
    if (side == BLACK)
      bestScore = -bestScore;
    return bestScore;
  }
  if (!gCounts[WHITE_INDEX]) {
    // White is wiped out!
    bestScore = kBestPossible;
    if (side == WHITE)
      bestScore = -kBestPossible;
    return bestScore;
  }

  if (!gCounts[BLACK_INDEX]) {
    // Black is wiped out!
    bestScore = kBestPossible;
    if (side == BLACK)
      bestScore = -kBestPossible;
    return bestScore;
  }

  if (depth <= 0) {
    if (stillOpen > kSolveThreshold &&
        gCounts[OPPONENT(side) >> COLOR_SHIFT]
        > kWipeOutExtension) {
      bestScore = Evaluate(side);
#ifdef USE_HASH
      bestMove = NULL;
#endif
      goto HashSave; //Terminal node
    }
  } else if (depth == 1 && stillOpen > kSolveThreshold) {
    t = Evaluate(side);
    if (t > beta + kFutilityScore)
      return t; // Futility cut-off
  }

#ifdef USE_HASH
  tableReply = NULL;
  pHashTable = &gHashTable[gHashValue & kHashTableMask];
  if (pHashTable->HashValue == gHashValue) {
    tableReply = pHashTable->BestMove;
    if (pHashTable->Depth >= depth) {
      if (pHashTable->Type == VALID) {
        if (pHashTable->Score > beta)
          alpha = beta;
        else if (pHashTable->Score > alpha)
          alpha = pHashTable->Score;
      } else if (pHashTable->Type == LOWER_BOUND) {
        if (pHashTable->Score > beta)
          return beta + 1;
      } else if (pHashTable->Type == UPPER_BOUND) {
        if (pHashTable->Score < alpha)
          return alpha - 1;
      }
      if (alpha > beta)
        return beta;
    }
  }
#endif

  // Abort?
  if (LMGetTicks() >= gAbortSearchTime) {
    gAborted = true;
    return 0;
  }

#ifdef USE_HASH
  bestMove = NULL;
#endif
  nextSide = OPPONENT(side);

  firstMove = gTreeEnd;
  generated = Generate(side);
  gMobility[gPly] = generated;

  if (!generated) { // no moves available
    if (passEndsGame) {
      bestScore = (gCounts[WHITE_INDEX] -
                   gCounts[BLACK_INDEX]) * kFinalDisk;
      if (side == BLACK)
        bestScore = -bestScore;
      return bestScore;
    }
#ifdef USE_HASH
    gHashValue ^= kSwitchSideHash;
#endif
    gIncScore = -gIncScore;
```

Royal Software Inc.

Clever Tools For Clever People!™

Over 50 Powerful Tools For HyperCard, SuperCard, OMO, and AppleScript! Including:

LiveCard: Run HyperCard over the Internet! **$99.95**
TCP Suite: FTP, SendEmail and GetURL **$99.95**
MegaWindow: External Windows were never easier! **$79.95**
QTCX: QuickTime conferencing **$99.95**
PrintReport: Print and Preview sophisticated reports! **$79.95**
CompileIt!: Hypertalk compiler - Create *any* code resource! **$49.95**

Satisfaction Guaranteed! A leader since 1984!

Get **10%** off your order [**20%** off orders over $200!] by mentioning this ad!

Royal Software, Inc.
300 Cedar Lane
Largo, FL 33770
800.888.7667 sales
813.581.6422 voice
813.559.0614 fax
sales@royalsoftware.com
http://www.royalsoftware.com

HEIZER

Including Heizer Software

Find out why Apple bundled one of our products with the latest release of HyperCard and CEO Amelio mentioned us by name at his keynote speech at MacWorld SF '97

```
    ++gPly;
      bestScore = -Search( beta, alpha,
                           nextSide, depth, true);
    --gPly;
    gIncScore = -gIncScore;
    ++depth;
    goto Searched;
  }

  to = firstMove;
  bestScore = -kInfinity;
#ifdef USE_HASH
  if (tableReply && *to != tableReply) {
    // Find tableReply in move list and move to front
    PSQUARE *p;
    for (p = to + 1; *p; ++p)
      if (*p == tableReply) {
        // Swap *p and *to
        *p = *to;
        *to = tableReply;
        break;
      }
  }
#endif

  saveIncScore = gIncScore;
#ifdef USE_HASH
  gHashValue ^= kSwitchSideHash;
  saveHashValue = gHashValue;
#endif

  do {
    if (!IS_BAD(**to) || // selectivity
#ifdef USE_HASH
          !bestMove ||
#endif
          gStartDepth - depth <= kFullDepthPlies ||
          stillOpen <= kSolveThreshold) {
      gIncScore = - (gIncScore + MakeMove(*to, side));
```

```
        ++gPly;
          t = -Search(-beta, -alpha, nextSide, depth-1,
false);
        --gPly;
        UnmakeMove();
        gIncScore = saveIncScore;
#ifdef USE_HASH
        gHashValue = saveHashValue;
#endif
        if (gAborted) {
#ifdef USE_HASH
          bestMove = NULL;
#endif
          break;
        }

        if (t > bestScore) {
#ifdef USE_HASH
          bestMove = *to;
#endif
          if (t > alpha) {
            if (t >= beta) {
              bestScore = t;
              break;
            }
            alpha = t;
          }
          bestScore = t;
        }
      }
      ++to;
    } while (*to);

Searched: ;
#ifdef USE_HASH
  gHashValue ^= kSwitchSideHash;
#endif
  gTreeEnd = firstMove;
#ifdef USE_HASH
  if (bestMove) {
#endif
HashSave: ;
#ifdef USE_HASH
    pHashTable = &gHashTable[gHashValue & kHashTableMask];
    if (pHashTable->Depth <= depth) {
      pHashTable->HashValue = gHashValue;
      pHashTable->Depth = depth;
      pHashTable->Score = bestScore;
      pHashTable->BestMove = bestMove;
      MY_ASSERT(!bestMove || IS_EMPTY(*bestMove));

      pHashTable->Type = VALID;
      if (bestScore <= oldAlpha) {
        pHashTable->Type = UPPER_BOUND;
      } else if (bestScore >= beta) {
        pHashTable->Type = LOWER_BOUND;
      }
    }
  }
#endif

  return bestScore;
}
```

Generate

```
#define ADD_MOVE(pSq)    *(gTreeEnd++) = pSq
long Generate(unsigned long side)
{
  PSQUARE *e, p, q, *afterCorners, *movesStart, lastBad;
  unsigned long enemy;
  long i, *pOffsets;

  enemy = OPPONENT(side);
  afterCorners = movesStart = gTreeEnd;
  lastBad = NULL;

  e = gEmptyAdj;
  i = gSizeEmptyAdj;
  while (i--) {
    p = *e;
    MY_ASSERT(IS_EMPTY(*p) && (*p & ADJACENCY));
    pOffsets = gOffsets;
    do {
      q = p + *pOffsets;
      if (*q & enemy) {
        do { // Skip through line of enemy disks
          q += *pOffsets;
        } while (*q & enemy);
        if (*q & side) { //Add square p to move list!
          // Bad? Try to put it on the end
          if (IS_BAD(*p)) {
            if (!lastBad) {
              lastBad = p;
              break;
            }
            ADD_MOVE(p);
            break;
          }
          if (!IS_CORNER(*p)) {
            // Add to end after corners
            ADD_MOVE(p);
            break;
          }
          // Corner: keep all corners on front
          if (afterCorners == gTreeEnd) { // All are corners!
            ADD_MOVE(p);
          } else {
            unsigned long *temp = *afterCorners;
            *afterCorners = p;
            ADD_MOVE(temp);
          }
          ++afterCorners;
          break;
        }
      }
    } while (*(++pOffsets));
    ++e;
  }
  if (lastBad) {
    ADD_MOVE(lastBad);
  }
  ADD_MOVE(NULL); // sentinel
  return (long)(gTreeEnd - movesStart) - 1;
}
```

MakeMove

```
// Makes the move for "side" on the "to" square
// Saves the move to gChanges for later undo'ing
// Returns the change in score relative to the moving side
SCORE MakeMove(PSQUARE to, unsigned long side)
{
  unsigned long z;
  PSQUARE q, p;
  long dir, offset;
  long flips = 0;
  SCORE score = 0;
  unsigned long enemy = OPPONENT(side);

  MY_ASSERT(IS_EMPTY(*to) && (*to & ADJACENCY));

#ifdef USE_HASH
  // Update hash for new disk
  if (side == BLACK)
    gHashValue ^= *(to + gBlackHashOffset);
  else
    gHashValue ^= *(to + gWhiteHashOffset);
#endif

  START_SAVE;

  // Do flipping, Update adjacency
  dir = 7;
  do {
    offset = gOffsets[dir];
    q = to + offset;
    if (IS_ON_BOARD(*q)) {
      if (IS_EMPTY(*q)) {
        if ( !(*q & ADJACENCY) ) { // First adjacent
          ADD_TO_EMPTYADJ(q);
        }
        score--; // New disk touches empty, which is bad
      } else if (*q & enemy) {
        // Skip through line of enemy disks
        p = q + offset;
        while (*p & enemy)
          p += offset;

        if (*p & side) { // Flip 'em
          p = q;
          do {
            // Flip disk at p
            ++flips;
            if (IS_EDGE(*p))
              score += kEdge;
            PUSH_SQ(p);
            score -= (COUNT_EMPTIES(*p) << 1);
                // x2 Empties counted for us now
            *p ^= COLOR_BITS; // Flip
#ifdef USE_HASH
            gHashValue ^= *(p + gFlipHashOffset);
                // Update hash for flipped disk
#endif
            p += offset;
          } while (*p & enemy);
          score++;
                // Fills in area around disk at q which is now ours
        } else {
          score--;
                // Fills in area around disk at enemy disk at q
        }
      } else { // *q & side
        score++;
                // Fills in area around our disk at q
      }
      z = OPP_DIR_BIT(dir);
      MY_ASSERT((*q & z) == 0L);
      *q |= z;
    }
  } while (dir--);

  PUSH_SQ(to);
  REMOVE_FROM_EMPTYADJ(to);
  *to |= side;

  PUSH(gCounts[WHITE_INDEX]);
  PUSH(gCounts[BLACK_INDEX]);
  gCounts[side >> COLOR_SHIFT] += flips + 1;
  gCounts[enemy >> COLOR_SHIFT] -= flips;

  if (IS_EDGE(*to))
    score += kEdge;

  return score;
}
```

UnmakeMove

```
void UnmakeMove()
{
  PSQUARE to, flipped, q;
  unsigned long z;
  long dir;

  // Restore disk counts
  gCounts[BLACK_INDEX] = POP;
  gCounts[WHITE_INDEX] = POP;

  // Replace to-disk
  to = (PSQUARE)POP;
  *to = POP;
  ADD_TO_EMPTYADJ(to);

  // Undo disk changes
  while (POP) {
    flipped = (PSQUARE)TOP;
    *flipped = POP;
  }
  // Update adjacency
  dir = 7;
  do {
    q = to + gOffsets[dir];
    z = OPP_DIR_BIT(dir);
    *q &= ~z; // Remove adjacency (if not already removed from disk changes)
    if ( IS_EMPTY(*q) && !(*q & ADJACENCY) ) { // First adjacent
      REMOVE_FROM_EMPTYADJ(q);
    }
  } while (dir--);
}
```

Initialize

```
void Initialize(long boardSize, void *privStorage)
{
  unsigned long *p, *q;
  long i, index, numRealSquares;
  char *ptr;
  unsigned long z;
#ifdef USE_HASH
  unsigned long r1, r2;
#endif

  kX       = -2 - boardSize * 5;
  kC       = -2 - boardSize * 4;
  kEdge    = 3 + boardSize / 8;
  kCorner  = 2 + boardSize * 13;

  gRealBoardSize = boardSize + 2;
  gNumOnSquares = boardSize * boardSize;
  numRealSquares = gRealBoardSize * gRealBoardSize;

  ptr = privStorage;

  gSquares = (unsigned long *)ptr;
  gOnBoardStart = gSquares + gRealBoardSize + 1;
  gOnBoardEnd   = gSquares + (gRealBoardSize *
(gRealBoardSize - 1) - 2);
  ptr += numRealSquares * 4;
  // worst case 66*66*4 = 17,424 bytes (~17K)

#ifdef USE_HASH
  gWhiteHashOffset = numRealSquares;
  gBlackHashOffset = numRealSquares * 2;
  gFlipHashOffset  = numRealSquares * 3;
  ptr += (numRealSquares * 3) * 4;
  // worst case 66*66*3*4 = 52,272 bytes (~51K)

  gHashTable = (SHash *)ptr;
  ptr += kHashTableSize * sizeof(SHash);
  // 32768*16 = 524,288 bytes (512K)
#endif
```

TCP/IP Scripting Addition *The Internet Scripting Solution*

The **TCP/IP Scripting Addition** allows you to quickly develop Internet client/server applications using AppleScript®. If you want to script with MacTCP ™ and Open Transport™, here's your solution!

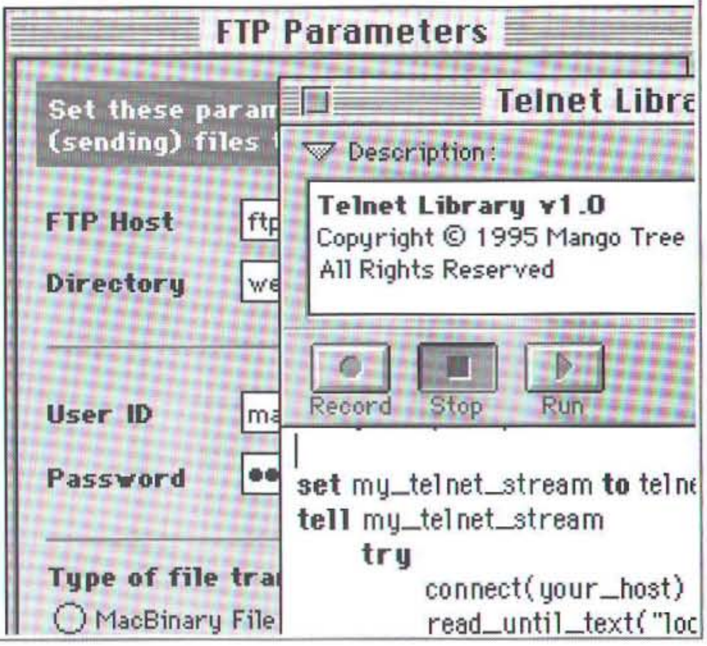
- Supports ScriptEditor, FaceSpan ™, and HyperCard™
- Build net–wise WebSTAR ™ CGI scripts and NetScape™ CCI scripts
- Sample scripts include FTP, Gopher, Telnet, Post Office, E–Mail and more
- Featured on the Apple® Internet Server

ONLY $49 SRP

Order now through the MacTech Mail Order Store at 805–494–9797 or other mail order stores

Mango Tree Software, Inc.
Box 1057 • Brookline, Massachusetts 02146
617–327–8663
http://www.mangotree.com/biz/mango

All trademarks are properties of their respective holders.
Contact Mango Tree Software for site licensing and redistribution information.
Copyright © 1996 Mango Tree Software, Inc.



Telnet Library v1.0
Copyright © 1995 Mango Tree
All Rights Reserved

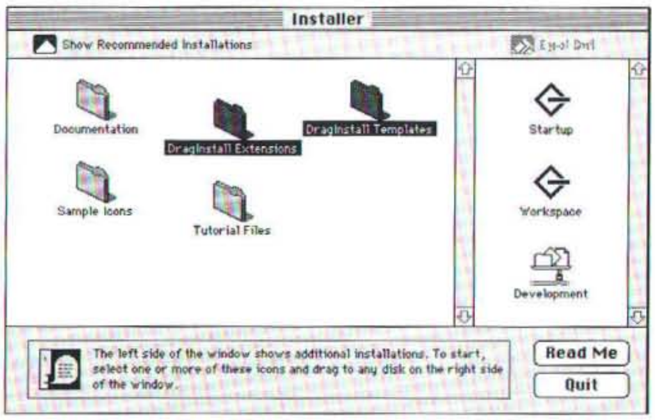
```
set my_telnet_stream to telne
tell my_telnet_stream
    try
        connect( your_host)
        read_until_text( "loc
```

```
gEmptyAdj = (PSQUARE *)ptr;
ptr += gNumOnSquares * 4;
// worst case 64*64*4 = 16,384 bytes (16K)

gChanges = (unsigned long *)ptr;
ptr += 65536; // e.g. 64 moves (deep) * 256 longs/move * 4 bytes/long
// 65,536 bytes (64K)

gMobility = (long *)ptr;
ptr += 1024; // e.g. 256 moves deep * 4 bytes/long
// 1,024 bytes (1K)

gCountZeros = (unsigned long *)ptr;
ptr += 256 * 4;
// 256*4 = 1,024 bytes (1K)

gTree = (PSQUARE *)ptr;
// Gets what's left, almost 400K!


// ** Calculate directional offsets
gOffsets[DIR_S]  = gRealBoardSize;
gOffsets[DIR_N]  = - gRealBoardSize;
gOffsets[DIR_SE] = gRealBoardSize + 1;
gOffsets[DIR_NW] = - gRealBoardSize - 1;
gOffsets[DIR_NE] = - gRealBoardSize + 1;
gOffsets[DIR_SW] = gRealBoardSize - 1;

// ** Borders
// Upper/Lower
p = gSquares;
q = gSquares + (gRealBoardSize * (gRealBoardSize - 1));
i = gRealBoardSize;
do {
  *(p++) = BORDER_BIT;
  *(q++) = BORDER_BIT;
} while (–i);
// Sides
p = gSquares + gRealBoardSize;
i = gRealBoardSize - 2;
do {
  *p = BORDER_BIT;
  p += (gRealBoardSize - 1);
  *(p++) = BORDER_BIT;
} while (–i);

// ** Edges
// Upper/Lower
p = gOnBoardStart;
q = gOnBoardEnd;
i = boardSize;
do {
  *(p++) = NW_BORDER | N_BORDER | NE_BORDER;
  *(q–) = SW_BORDER | S_BORDER | SE_BORDER;
} while (–i);

// Sides
p = gOnBoardStart;
q = gOnBoardEnd;
i = boardSize;
do {
  *p |= NW_BORDER | W_BORDER | SW_BORDER;
  *q |= NE_BORDER | E_BORDER | SE_BORDER;
  p += gRealBoardSize;
  q -= gRealBoardSize;
} while (–i);

// ** Starting configuration
// Set up initial disks and adjacent empty squares
// "On your first move, you should initialize the board
// with white tiles at (row,col) = (boardSize/2-1,boardSize/2-1) and
// (boardSize/2,boardSize/2), and black tiles at (boardSize/2-1,boardSize/2)
// and (boardSize/2,boardSize/2-1)"
gCounts[WHITE_INDEX] = gCounts[BLACK_INDEX] = 2;

gSizeEmptyAdj = 12;
i = boardSize >> 1; // x2
index = XY2INDEX(i-1, i-1);

p = &gSquares[index];
*p = SE; gEmptyAdj[0] = p;
*(++p) = EMPTYADJ_BIT(1) | S | SE; gEmptyAdj[1] = p;
*(++p) = EMPTYADJ_BIT(2) | SW | S; gEmptyAdj[2] = p;
*(++p) = EMPTYADJ_BIT(3) | SW; gEmptyAdj[3] = p;
```

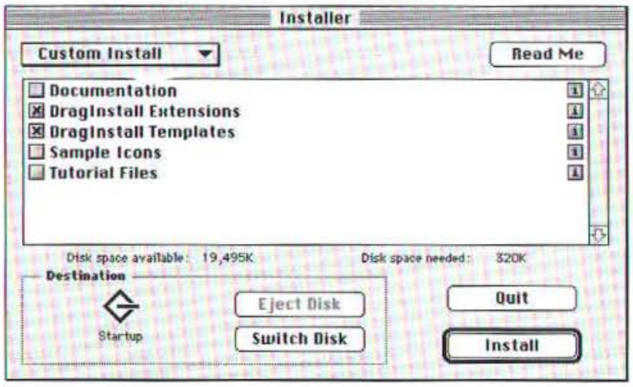
```
  p += gRealBoardSize - 3;
  *p = EMPTYADJ_BIT(4) | E | SE; gEmptyAdj[4] = p;
  *(++p) = WHITE | S | SE | E;
  *(++p) = BLACK | W | SW | S;
  *(++p) = EMPTYADJ_BIT(5) | W | SW; gEmptyAdj[5] = p;

  p += gRealBoardSize - 3;
  *p = EMPTYADJ_BIT(6) | E | NE; gEmptyAdj[6] = p;
  *(++p) = BLACK | N | NE | E;
  *(++p) = WHITE | W | NW | N;
  *(++p) = EMPTYADJ_BIT(7) | W | NW; gEmptyAdj[7] = p;

  p += gRealBoardSize - 3;
  *p = EMPTYADJ_BIT(8) | NE; gEmptyAdj[8] = p;
  *(++p) = EMPTYADJ_BIT(9) | N | NE; gEmptyAdj[9] = p;
  *(++p) = EMPTYADJ_BIT(10) | NW | N; gEmptyAdj[10] = p;
  *(++p) = EMPTYADJ_BIT(11) | NW; gEmptyAdj[11] = p;

  gNWCorner = gOnBoardStart;
  gNWC1 = gNWCorner + 1; *gNWC1 |= BAD_BIT;
  gNWC2 = gNWCorner + gRealBoardSize; *gNWC2 |= BAD_BIT;
  gNWX = gNWC2 + 1; *gNWX |= BAD_BIT;

  gNECorner = gNWCorner + boardSize - 1;
  gNEC1 = gNECorner - 1; *gNEC1 |= BAD_BIT;
  gNEC2 = gNECorner + gRealBoardSize; *gNEC2 |= BAD_BIT;
  gNEX = gNEC2 - 1; *gNEX |= BAD_BIT;

  gSWCorner = gOnBoardEnd - boardSize + 1;
  gSWC1 = gSWCorner + 1; *gSWC1 |= BAD_BIT;
  gSWC2 = gSWCorner - gRealBoardSize; *gSWC2 |= BAD_BIT;
  gSWX = gSWC2 + 1; *gSWX |= BAD_BIT;

  gSECorner = gOnBoardEnd;
  gSEC1 = gSECorner - 1; *gSEC1 |= BAD_BIT;
  gSEC2 = gSECorner - gRealBoardSize; *gSEC2 |= BAD_BIT;
  gSEX = gSEC2 - 1; *gSEX |= BAD_BIT;

  // Precalculate gCountZeros
  // (Could have had the compiler fill these in, but I'm not
  // THAT desperate for speed)
  for (z=0; z<256; ++z) {
    gCountZeros[z] =
      8 - (z & 1) - ((z>>1) & 1) - ((z>>2) & 1) - ((z>>3) &
1) -
      ((z>>4) & 1) - ((z>>5) & 1) - ((z>>6) & 1) - ((z>>7) &
1);
  }

#ifdef USE_HASH
  gHashValue = 0xFFFFFFFF;

  // Initialize gHashKeys
  srand(0x1234); //srand(time(NULL));
  for (i=0; i<numRealSquares; ++i) {
    r1 = rand() + ((unsigned long)rand() << 16);
    r2 = rand() + ((unsigned long)rand() << 16);
    gSquares[gWhiteHashOffset + i] = r1;
    gSquares[gBlackHashOffset + i] = r2;
    gSquares[gFlipHashOffset  + i] = r1 ^ r2;
  }

  // Clear gHashTable
  {
    SHash *pHashTable = gHashTable;

    i = kHashTableSize - 1;
    do {
      pHashTable->HashValue = 0;
      pHashTable->Depth = -100;
      pHashTable->BestMove = NULL;
      pHashTable->Type = INVALID;
      pHashTable->Score = 0;
      ++pHashTable;
    } while (i--);
  }
#endif

  gIncScore = 0;
}
```

MT

Guess which installer was built by DragInstall.®



The left side of the window shows additional installations. To start, select one or more of these icons and drag to any disk on the right side of the window.



☐ Documentation
☒ DragInstall Extensions
☒ DragInstall Templates
☐ Sample Icons
☐ Tutorial Files

Disk space available: 19,495K Disk space needed: 320K

Guess again.

If you guessed that the first installer was built by DragInstall, you're only half right. Now with DragInstall 3.0, you can build *both* of these installers from a *single* script file!

For more information about DragInstall and a free demo, visit our web site at **http://www.sauers.com/draginstall** or give us a call at **1-800-890-9880**.

by Dave Mark, ©1997 by Metrowerks, Inc., all rights reserved.

Greg Galanos and the Mac Developer's Roadmap

This month is, perhaps, one of the most important in Apple's history. Although Apple has shown some spark in recent months with the release of some top notch products, the key to their success clearly lies in the timely release of Rhapsody and Rhapsody related tools. With that in mind, this month's interview is with Greg Galanos, Metrowerks' founder, President, and Chief Technology Officer. Greg presents his views of Macintosh development à la CodeWarrior as we make our way into Rhapsody.

Dave: Let's start off with an overview. What changes do you see coming down the pike for CodeWarrior? Will the versions of CodeWarrior retain a unified look on different hosts?

Greg: No matter where you program, you will have essentially the same set of tools to do your work. The host platforms that we are committing to and their order of importance (for our Mac users) are: MacOS, Rhapsody, Windows95, WindowsNT, BeOS, Solaris, SGI and HP. We are striving to create one powerful, integrated development platform regardless of the host.

We're targeting the creation and deployment of a complete development platform that includes project management, project builds, editing, view editing, resource editing and codegen.

The new project manager is ready for prime-time and incorporates many new features including sub-projects, multi-target builds, a new build system, new searching algorithms and the new CodeWarrior look and feel. The CodeWarrior VCS plug-in for plugging alternative source-code control systems is now being supported by CodeManager from Metrowerks, Voodoo from Uni Software Plus, and various shareware and freeware versions of Projector.

We intend to move ahead with a tools driver architecture that will make support of multiple tools plug-ins to be better maintainable from our own internal engineering perspective. We will be deploying a new generation of the CodeWarrior plug-in API that allows us to take into account endian issues between different host platforms (necessary to account for little-endian NT and little-endian Rhapsody), and simplifies the task of porting command-line tools to be CodeWarrior plug-ins.

As far as RAD is concerned, the objective for the RAD part of CodeWarrior will be to support PowerPlant for C++ on MacOS, MFC for C++ on Windows, the OPENSTEP and Rhapsody APIs with Objective-C on Rhapsody, and the conglomerate of APIs for Java (AWT, IFC, JavaBeans, AFC) on all platforms. We will implement fine-grained code-generation to support these technologies.

On Debugging, the debugger will be integrated into CodeWarrior and incorporate a plug-in architecture that allows multi-target and two-machine debugging from within CodeWarrior.

The upside to all this is that all the components of CodeWarrior will be integrated — you will be able to manage your work flow much more efficiently, and transitions between CodeWarrior components will be much smoother from coding, to builds, to debugging, to browsing, to user interface design and back.

Dave: What's the time frame for these changes?

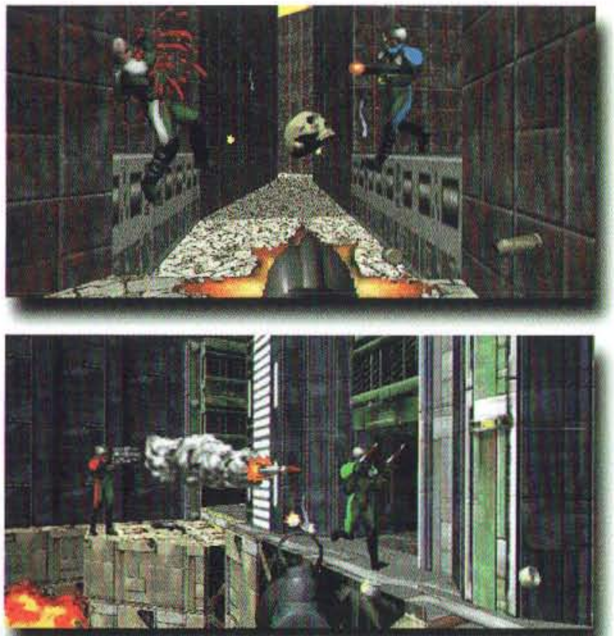
Greg: Our plan is to deploy these technologies within CodeWarrior over the next 3 to 6 releases of the CodeWarrior SDK, starting with the just released CW12.

Behind every good Mac app...

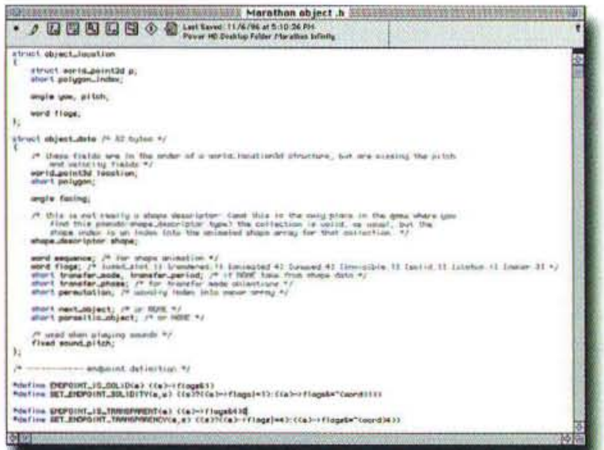
Every job requires the proper tool. When it comes to crafting good code, **BBEdit 4.0** is the **Swiss Army Chainsaw** of text editors.

"If you have to spend time with text on the Mac, you should use BBedit."
-- MacWeek, September 30, 1996

- **Compare** source files and **apply differences**
- **Open From** and **Save To FTP**
- **Search** and **replace** multiple files using **Grep**
- Navigate source files with integrated **PopupFuncs™** technology
- **Browse** project documents



Bungie Software's **Marathon Infinity**, created using **BBEdit**.



Marathon object .h
Last Saved: 11/6/96 at 5:10:26 PM
Power HD:Desktop Folder:Marathon Infinity

```
struct object_location
{
	struct world_point3d p;
	short polygon_index;

	angle yaw, pitch;

	word flags;
};

struct object_data /* 32 bytes */
{
	/* these fields are in the order of a world_location3d structure, but are missing the pitch
		and velocity fields */
	world_point3d location;
	short polygon;

	angle facing;

	/* this is not really a shape descriptor: (and this is the only place in the game where you
		find this pseudo-shape_descriptor type) the collection is valid, as usual, but the
		shape index is an index into the animated shape array for that collection. */
	shape_descriptor shape;

	word sequence; /* for shape animation */
	word flags; /* used_slot.1| rendered.1| animated.4| unused.4| invisible.1| solid.1| status.1| owner.3 */
	short transfer_mode, transfer_period; /* if NONE take from shape data */
	short transfer_phase; /* for transfer mode animations */
	short permutation; /* usually index into owner array */

	short next_object; /* or NONE */
	short parasitic_object; /* or NONE */

	/* used when playing sounds */
	fixed sound_pitch;
};

/* ---------- endpoint definition */

#define ENDPOINT_IS_SOLID(e) ((e)->flags&1)
#define SET_ENDPOINT_SOLIDITY(e,s) ((s)?((e)->flags|=1):((e)->flags&=~(word)1))

#define ENDPOINT_IS_TRANSPARENT(e) ((e)->flags&4)
#define SET_ENDPOINT_TRANSPARENCY(e,s) ((s)?((e)->flags|=4):((e)->flags&=~(word)4))
```

Source code from **Marathon Infinity** displayed in **BBEdit**.

...is a great text editor.

BBEdit 4.0 MacUser MacWEEK score card

http://www.barebones.com/chainsaw

To order, call 617-778-3100

Bare Bones Software, Inc.
P.O. Box 1048, Bedford, MA 01730 • main 617-778-3100 • fax 617-778-3111

Dave: You often speak about the "road to Rhapsody" and your desire to provide a road map for developers. Can you clarify that a bit?

Greg: The road to Rhapsody is more like a six-lane highway. This highway has to carry the CodeWarrior Platform and Tools as well as the entire Mac community from where we are now across to Rhapsody. Different technologies need to be implemented in each lane and not all technologies move at the same speed. I see the lanes as:

1. CodeWarrior compilers and linkers.
2. Current NeXT compilers and linkers.
3. CodeWarrior Debugger technology.
4. CodeWarrior Environment.
5. CodeWarrior Latitude.
6. Metrowerks PowerPlant.

Dave: How do you see the evolution of the CodeWarrior compilers and linkers?

Greg: The CodeWarrior compilers and linkers will be modified to allow our customers to use their language of choice, and to use the current system-preferred language, Objective-C.

In order to do this, we're essentially taking a path analogous to the path chosen for SOM. When we implemented support for direct to SOM in the CodeWarrior C++ compiler, you would essentially derive from a SOM object in your standard C++ source code and the compiler front-end would take care of generating the correct runtime bindings so that the SOM object could converse with the system-level SOM runtime. This was called direct to SOM or dtS and it allowed us to bypass using SOM IDL (Interface Definition Language).

We'll be taking the same approach to implement Objective-C runtime support. This means you will derive from an Objective-C object in your C++ or Object Pascal program and the compilers will generate the proper runtime bindings for the Objective-C runtime.

In the above case, Apple will need to provide the C++ and Pascal headers for Rhapsody that contain the entry point in the Objective-C system runtime for this to work, but at this time it is our understanding that either Apple or Metrowerks will find a way to make this happen.

The conclusion here: use your current language of choice and bind to the Rhapsody system APIs.

We also want to provide people with the possibility of adding Objective-C syntax to either their C or their C++ code. This will be done through the use of a front-end language extension switch that will tell the compiler to accept the proper Objective-C syntax construct and generate the proper code for it.

So, that is what we're doing from the compiler standpoint. It's interesting to point out that this work will be used by both the PowerPC and Intel back-ends which will allow us to also implement support for Rhapsody running on Intel as this seems to be a direction that Apple will continue to support.

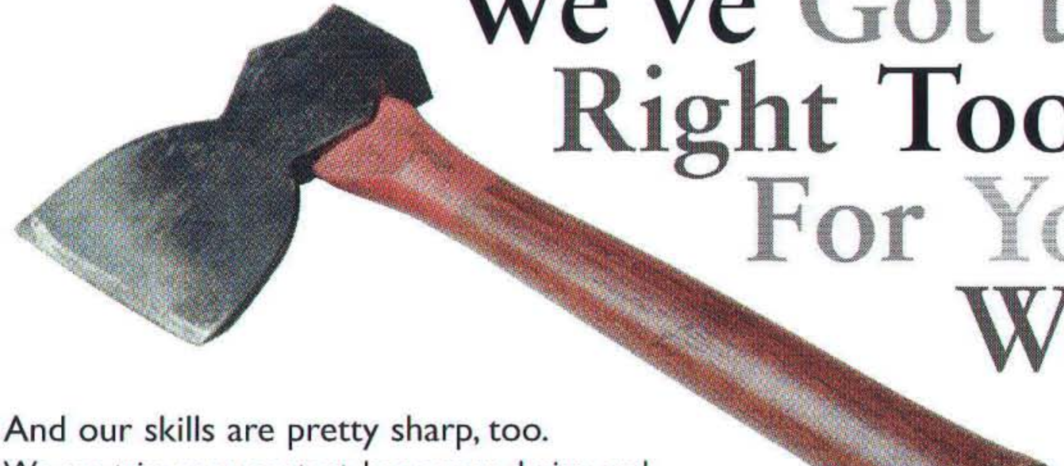
Changing the compilers is great and fills up half the lane but we need to go further and generate the system executable format. On Rhapsody, the underpinning is the Mach operating system version 2.5 or higher, so we need to implement support for native linking of this format. In order to do this, we need to support Mach-o, the native executable format on Rhapsody. This is well defined on Intel where the Mach kernel is already operational (under OPENSTEP). By the time you read this, it will also have been well defined on PowerPC as we work closely with Apple to make this happen.

Oh, I forgot to mention the symbolic debugging format... It looks like we are going to standardize on DWARF (Debug With Arbitrary Records Format). DWARF is an industry accepted standard debugging format used extensively on SVR4, BSD4.3 and embedded operating systems. It is a processor and platform independent data format, rich enough to define all symbolic necessary for full source-level debugging. It is also language independent.

Now, let's complicate matters some. What about OPENSTEP running on NT? Well, it uses the standard Microsoft COFF format executable, and this is essentially fixed up at runtime by the OPENSTEP Objective-C runtime loader so that the executable can run and connect into the OPENSTEP runtime. Don't ask me how they do it, those NeXT guys made magic. However, given that our Windows95/NT compilers/linkers support Microsoft COFF, then the NeXT magic should work with our NT compilers too, which means that with the CodeWarrior toolset, you should be able to build binaries not only for Rhapsody on PowerPC and Intel running on top of the Mach kernel, but also for OPENSTEP running on top of NT. I'm not even going to start talking about OPENSTEP on Solaris because it's starting to make my head spin like Meryl Streep's in *Death Becomes Her.*

Dave: How do the current NeXT compilers and linkers fit into all this?

Greg: OPENSTEP's current tools architecture leverages the important work done by the Free Software Foundation in their support of the GNU compiler and linker tool set. The GNU compilers and linkers are well known for the quality of their generated code albeit in the form of compilers and linkers that are definitely not on par with CodeWarrior compilers from a strict 'time to market' (read high speed) standpoint. Nevertheless, these compilers, which form the basis of the Objective-C compiler technology used on OPENSTEP, are used today by most if not all OPENSTEP developers. They build the code that runs on the platform today. We at Metrowerks run into the GNU



We've Got the Right Tools For Your Web Site!

And our skills are pretty sharp, too.
We can trim your content, hone your design, and implement a Web site that's...well...a cut above the rest.

With today's chopped budgets and pared-down schedules, you need JointSolutions Marketing. We have the experience, and the tools, to make sure your World Wide Web pages aren't...you know...dull.

We even have a Web server, so site maintenance and updates don't slice into your work time.

Call us today and we'll take a whack at your Web needs.

JointSolutions Marketing
Tel: (408) 471-1500
E-mail: info@jointsolutions.com
http://www.jointsolutions.com

---

compiler architecture all over the place. For example, whenever first silicon for a new processor is brought up, someone usually retargets a GNU compiler to support the chip. This allows early users to quickly approach the new silicon.

Because they are being used for the current incarnation of Rhapsody and also for silicon bring up, it is in Metrowerks' best interests to make sure that these compilers and linkers can work properly within the CodeWarrior Platform. For these reasons, we've decided to fund the development of GNU compiler and linker plug-ins to the CodeWarrior IDE. Once the plug-ins are developed, the plug-in code, based on the CodeWarrior Plug-in Tools API, will be given back to the Free Software Foundation so that anyone using the GNU technology can build a GNU compiler or linker as a CodeWarrior plug-in.

This meets three objectives; First, it allows current CodeWarrior customers to use the current Rhapsody compilers and Rhapsody linker to support seed versions of Rhapsody or current versions of OPENSTEP from within the CodeWarrior environment;

Second, as we move the CodeWarrior Platform to Rhapsody, it will allow OPENSTEP developers to use the CodeWarrior Development Environment as the back-end build system to Interface Builder as an optional build system to NeXT's (er, I mean Apple's) Project Builder on OPENSTEP. This allows the merging of two state of the art technologies — CodeWarrior for builds, Interface Builder for RAD — to interoperate seamlessly (or that's the plan).

Finally, it allows us to transition GNU technology users to CodeWarrior in the embedded systems market as chip volume for new silicon increases and customers start looking for high-volume, low-cost, off-the-shelf development tools in order to support a new or existing ISV developer base on a new device.

By the time you read this we may have figured out what we're going to do regarding debugging for GNU, but our game plan right now is to figure out how to use our own source-level debugging technology with the GNU compiler technology.

Finally, with respect to current OPENSTEP tools, one thing missing is to build the OPENSTEP/Rhapsody linker as a CodeWarrior plug-in. We're working closely with Apple to achieve this goal. By the time you read this, I assume we'll also have figured out how to get the Apple technology such as the Linker to you efficiently.

Dave: What are your plans for Java?

Greg: Java clearly is here to stay and Metrowerks is investing aggressively in the areas of Java technology that require world-class tools: JITs, Java compiler, RAD, and VMs.

It's Macintosh Accounting At Its Best

FLEXIBLE

FlexWare is designed to give you almost unlimited possibilities as your company grows.

POWERFUL

FlexWare's accounting features and reporting capabilities give you information you'll rely on daily to make smart decisions.

FAST

Large businesses consistently choose FlexWare because of its exceptional speed on network and client/server systems.

Meet your clients' needs with the FlexWare Development System. FlexWare is an integrated, modular, *MacUser* Recommends award-winning accounting program that you can modify and customize to suit each of your clients' requirements. Take advantage of FlexWare's top performance and adaptability. It's flexible. It's powerful. It's fast. Call the FlexWare Sales Department at 800-447-5700 for information on the FlexWare Development System.

FlexWare™
ACCOUNTING

STATE OF THE ART
ACCOUNTING SOFTWARE

8211 Sierra College Blvd., Suite 440, Roseville, CA 95661-9406 • ✆ 916-791-7730 • FAX 916-791-5525 • 1-800-447-5700

Metrowerks JIT technology is based on a retargetable compiler technology that allows us to retarget the JIT for different processors. Currently, Metrowerks has JITs available for PowerPC and 68K MacOS. The fact that these JITs are developed and tested on mklinux also means that they are quite platform independent and can be used both for Rhapsody and for embedded 68K and PowerPC targets with some modifications. Our first objective with respect to JITs was to create a robust implementation and we have done so.

Our follow-up objective is to provide better optimizations as we are doing this. Every iteration of a CodeWarrior JIT has seen increased performance. We are weighing optimizations versus real-time performance as optimizations are clearly a two-edged sword when you're JITing code. The more optimizations you put into the JIT, the slower it JITs the applet. It's very important to balance optimizations with runtime performance and we're making sure we don't optimize our JIT to the point where it takes longer to translate the code than it does to interpret the code.

Additionally, we are building JITs for embedded targets. One of the first is for NEC Electronics V-800 series processor, a RISC processor with approximately 20 times the performance of the 68k series from Motorola, but at around the same price point. The V800 processor is the building block for NEC's systems-level offering for digital television (satellite receivers, Internet-ready television, digital audio/video sub-systems).

We're also working closely with Sun Microsystems, Apple and Microsoft to implement and support the JIT APIs from these companies, so that JITs can be interoperated with different VMs.

Dave: How about a native Java compiler?

Greg: Java is a real interesting beast in that even though it is platform and processor neutral, different technologies need to be implemented in order for it to be efficient in both execution and memory footprint. It challenges the developer of tools in that a wide variety of options are necessary depending on the runtime characteristics of the target. For instance, if the target is a desktop machine with loads of RAM, either JIT or native binary compilation is an option. However, both of the above solutions increase memory usage dramatically (an order of magnitude) for the final executable, compared to just interpreting the code with the VM interpreter.

We're making sure our customers have all the options available in CodeWarrior. One of the things we'll be doing this year is developing our own fully integrated Java compiler. This compiler, comprised of a front-end and a back-end, will be fully merged into the CodeWarrior compiler technology base. The front-end will generate code to the CodeWarrior IR (intermediate memory resident representation). The Java back-end will generate the bytecodes. The advantage of generating for the IR is two-fold. First, it allows us to leverage the optimizers used by current compilers (C/C++ and Pascal) on the IR. This in turn allows the Java back-end to generate a 'better' byte-code sequence. Look at this as generating better Java byte-codes, whether they are JITed or run on the VM, their optimized generation will make them run more efficiently (at least that's the objective).

Of course, the fact that the compiler will be written in ANSI C, as all of the CodeWarrior compilers are today, will also give customers a screaming implementation of the Java compiler as fast if not faster, given that it's a simpler language, than our C/C++ compiler.

Secondly, given that we are generating the CodeWarrior IR, we will also be able to use Metrowerks back-end technology for specific processors (PowerPC,68K,x86,MIPS,V800) to generate a pure native binary, one that will not require JITing and will run on the target processor. The caveat here is that we still need to figure out how to provide system services provided by the VM (such as garbage collection, etc.) to the native binary. We're working out the details of this with Apple and Microsoft for the two desktop platforms we support. For the time being, we'll be calling the VM for the service where necessary, until a standard "VM-native services" library is implemented by Apple and Microsoft.

We're also continuing to invest in the Sun Java compiler as the currently shipping compiler for CodeWarrior. Currently, the compiler has already been J2Ced. This essentially takes the current compiler, translates it to a C representation, then the CodeWarrior C compiler compiles it. It gives us a "native" compiler that generates byte-code, but that is still a cousin of the original Sun compiler. Of course, we've also gone to great lengths to architect the Sun compiler so that it looks and acts like a normal CodeWarrior plug-in. We'll be implementing an IR-generating pass in the Sun compiler that will allow us to generate the CodeWarrior IT directly, in order to start solving the runtime issues that arise from native code generation while we build the full CodeWarrior Java compiler.

So, in conclusion on Java compiler technology (JIT and Static compilation), we are implementing dynamic compilation (JIT), better more optimized compilation (Java compiler), static compilation (Java front-end to processor-target back-ends) and enhanced static compilation (other language front-ends to PicoJava processor target). Interesting, important stuff — the baseline for our tools infrastructure for Java, critical in the years to come as Java moves from early adoption to widespread use.

Dave: What about Java VM support?

Greg: We got into the VM business early in 1995 when it was clear that we needed a VM, and we needed it for our customers earlier than Apple was going to provide a VM. The Metrowerks VM, a derivative of the Sun VM, developed under license, is used today in Internet Explorer on MacOS and we've just relaxed our licensing restrictions in our CodeWarrior software license so that any of our customers can ship the VM bundled with their Java applications. We developed two optimized byte-code interpreters, one for PPC the other for 68K. The interpreters essentially accelerate about 150 byte-codes, and translate into a VM that is about 60% faster than a standard VM without JITs. We've just licensed this technology to Apple Computer for incorporation into Apple's VM. We'll also be shipping Apple's VM as an option to the Metrowerks VM as part of any CodeWarrior Java toolkit (Gold, Academic, Discover Java).

As soon as Apple and Microsoft agree on support for COM in Apple VMs, we will move to transition our customers to one VM on MacOS, provided to all customers by Apple as a baseline system service. Clearly, our goal is to exit the VM business as this is really a system vendor area where we do not provide clear, added value long-term. On Windows95 and WindowsNT, we will essentially ship the Microsoft VM and x86 JIT. I'm assuming that there will be a standardization effort on MacOS and Rhapsody that will allow us to do the same thing on Apple platforms.

We'll be happy to work with Apple on providing the JIT technology as we have provided the OBCI technology and are working towards this goal with Apple and Microsoft.

Dave: And Java RAD?

Greg: Clearly, our first efforts to support Java from within Constructor have been quite successful, but we believe we definitely need to go much further in the support of visual development in Java. As such, as part of our RAD implementation in CodeWarrior, we will be implementing complete support for Java visual development from within CodeWarrior, support for the development of the user interface in a visual fashion, with fine-grained code-generation, object wiring, and 'instant run' to provide immediate feedback to the developer of a Java application. Coupling this with a world-class development environment and build system that supports multiple languages will provide a world-class, very powerful solution regardless of platform. The RAD for Java support features will be rolled out in CodeWarrior in a stepwise fashion as part of each CodeWarrior SDK release as other technologies for C/C++ and Pascal have been rolled out over the last 12 releases. (12 releases in 4 years, I must be getting old.)

Dave: What's the story with CodeWarrior's debugging technology?

Greg: As I said, we are working on a merger of the IDE and the host debugger technology to make the development experience continuous. The current CodeWarrior debugger will be integrated into the development environment, so you will be able to edit, compile, and debug all within the same environment. This means all the functionality of the IDE will be available when you are debugging, and vice versa. For example, you will be able to set breakpoints in your code while you are editing, and you will be able to click on a symbol in the source you are debugging and make use of the pop-up that appears (jump to a symbol's definition, for example).

We are also adding support for two machine debugging in CodeWarrior 12. The CW 12 MetroNub now supports both single machine debugging and two machine debugging via TCP/IP. Now, you can debug your Mac applications with the debugger running on the same machine, on a second Mac, or from your Windows box. You can also use the same debugger to debug 68K, PowerPC, x86 and Java, all simultaneously.

Supporting Rhapsody debugging is an interesting beast because essentially we're talking about a multi-tasking, preemptive, memory-protected kernel and (you guessed it!) a new range of services to support on that kernel.

Trivia-start — we did do some of this work for Copland (you remember that thing, dimly in the past, like last May). Actually, all humor aside, some of you may recall that Metrowerks shipped the DR1 of a CodeWarrior debugger that actually debugged code running on Copland, back in May 1996. That piece of code was probably the only piece of commercial software ever shipped for Copland — Trivia-end.

Rhapsody uses the Mach kernel, which is well implemented as originally defined by Carnegie-Mellon. We are taking a different approach to debugging on this new operating system, actually peering into Apple's past and bringing some very interesting technology to bear on the problem. From where you might ask? Well, from the work done on Hoops by Taligent.

A very neat debug server architecture was developed back then to run on top of AIX and to provide debug services to the Hoops environment. As part of our relationship and development agreements with Apple, we have licensed that technology and are porting it to Mach. This debug server will allow both full, one machine debugging as well as two-machine debugging. Then we need to change the host debugger to support two machine debugging, but a lot of this original work was already done to support target debugging (MacOS hosted CW to NT, MacOS-hosted CW to PlayStation).

So hopefully, you'll see a very clean, modern architecture for Rhapsody debugging that will work for pretty well any platform on which Rhapsody runs. Of course, MetroNub will continue to support MacOS multi-language debugging and the Hoops nub will do so for the new OS. CodeWarrior's host debuggers, whether hosted on MacOS, Rhapsody, Win95 or WinNT or a UNIX platform will also be able to talk to this new debug server via TCP/IP.

Dave: How will Latitude and Equal fit into the picture?

Greg: Metrowerks recently acquired the assets of the Latitude Group which was essentially two products: Latitude, a well-architected porting library and Equal, a 68K emulator, linked with the Latitude library. We may use the 68K emulator in our embedded products at some later date, but right now, our immediate focus is on Latitude.

The really interesting thing about Latitude is that it was architected from the ground up to support the emulator. As such, it has a distinct design and has proven very robust in porting applications to UNIX systems, specifically Solaris, SGI and HP. In particular, Adobe used Latitude to port and run Premiere on SGI and Photoshop on Solaris. So we're talking about some very robust technology that is in current commercial use with two very demanding applications. Currently, The Latitude Cross-Platform Development Kit runs on the following UNIX RISC workstations:

- Sun Microsystems SPARC, or compatible, running Solaris 2.3 or higher.
- Silicon Graphics Indigo or Indy workstations, running IRIX 5.2 or higher.
- Hewlett Packard Series 700 workstations, running HPUX 9.03 or higher.

Latitude is a portable implementation of Macintosh System 7 application programming interfaces (APIs), as documented in Apple's *Inside Macintosh*. The Latitude Technology forms the core of the Latitude Cross-Platform Development Kit. The Kit is a virtual porting system that allows source code for a Macintosh application to be compiled with little or no modification on a UNIX platform, resulting in a native UNIX application.

All applications created with Latitude adopt the look and feel of the destination platform's GUI. The target GUI (such as OPEN LOOK or Motif) provides menus, dialog boxes, icons, scroll bars, and the like, but the application's content region retains the look and feel of the original Macintosh program. Special Latitude provided functions allow developers to tailor the content regions so that they too can have the target GUI look and feel. Latitude uses industry-standard interfaces including ANSI C, PostScript, and POSIX.

The Latitude technology performs the functions of the Macintosh API through a shareable set of program libraries. Developed for accuracy and portability, these libraries accurately reflect the behavior of most of the Macintosh System 7 toolbox. Well-behaved Macintosh applications built with Latitude run with little to no UNIX porting or customization.

Written in ANSI C for POSIX-compliant platforms that support X11, the Latitude technology is more that 95%

independent of the specific UNIX implementation, X11 libraries, and GUI under consideration.

Dave: How about the Latitude bridge to Rhapsody?

Greg: As a result of the unique architecture of the Latitude library, support of a new target like Rhapsody requires the retargeting of around 100 Latitude calls, about 50 user interface calls and about 50 imaging calls. The reason why we can move 2000 or so Mac calls to a new target so quickly is that the 2000 calls all first target the Latitude portability layer which abstracts out a lot of the target support and only requires a subset of calls to be implemented for a new target.

Given this, we believe that Latitude can provide a very strong portability solution for getting Mac apps up on Rhapsody (or other supported UNIX platforms) quickly, providing the Mac developer with the opportunity to execute a quick port to Rhapsody of their current application, call it a point release, and then to subsequently add in native service support for the target operating system.

A Macintosh application that binds to Latitude will acquire the look and feel of a Rhapsody application as well as protected-memory. Additional features, such as multi-tasking, can then be added through direct mixing of calls, some handled by the Latitude library, others directly by the native Rhapsody APIs.

As usual, it's only when we finish the implementation of the Rhapsody target and use it ourselves that we'll know if it's a hit or not. We think it will be, given our discussions with Adobe and Apple engineering. So, the first thing we'll do is eat our own dog food: port CodeWarrior using Latitude. Hopefully, by the time you read this we'll have something to show you.

We're hopeful we can ship the CodeWarrior Latitude SDK later this year, around the same time we ship the early releases of our Rhapsody tools.

Dave: How efficient will this approach be?

Greg: The best way to answer that is to look at apps that use a porting library. First, we use the Mac2Win library from Altura software to port CodeWarrior to Windows 95 and Windows NT. Given that the underlying file systems are native, and that the window, menu and dialog DLLs on Windows 95 and Windows NT are pretty efficient, we see no material slowdown in the Windows-hosted version of CodeWarrior. Given that the current Mac file system is emulated on Power Macintosh, we actually see a faster response time on CodeWarrior for Windows and no apparent performance difference between CodeWarrior using the Mac2Win library and native Windows apps.

As far as the Adobe products are concerned, they run much faster on SGI and Solaris workstations, the result both on the underlying UNIX kernels and the power of the workstations. So, we see no immediate need for concern here. Additionally, given that you can mix Latitude and native calls, if you experience a slowdown in a time-critical routine, you can just call the native API for that specific routine. So, instead of rewriting it all, you fine tune the parts that need fine tuning once you're up and running on Rhapsody.

Of course, we view Latitude as a transitional technology. It should be used for bring up on Rhapsody, and will not necessarily track all new Rhapsody APIs as they become available from Apple. It will certainly be worthwhile for a developer to implement the second generation products for Rhapsody directly to the native APIs or to write directly to Rhapsody for a new product.

Dave: How will Latitude be packaged?

Greg: The Latitude SDK, which will include all the sources to the Latitude library as well as all the supported targets (Rhapsody, Solaris, SGI, HP) in source will carry a retail price of $399. It will follow the CodeWarrior subscription model and be updated three times a year. Current CodeWarrior Academic discounts will also apply.

The new Rhapsody tools (CodeWarrior IDE and all the tools) will be rolled directly into the CodeWarrior Gold and Academic SDKs, so that all current subscribers will get the tools when they are available at no additional cost.

So, the Mac developer, armed with CodeWarrior Gold or Academic and CodeWarrior Latitude, with a little sweat equity (i.e. about thirty 'burning the midnight oil' nights) can move to Rhapsody fast, and we mean fast, without completely rewriting their application.

CodeWarrior Rhapsody will use the Latitude technology to leap to the new platform and be rolled into the CodeWarrior SDK. Once CodeWarrior is up and running on the platform, we'll continue adding new features in the product that take advantage of the native APIs. We're also taking a real hard look at InterfaceBuilder to see what we need in the environment to offer comparable features. Of course, developers will also be able to use InterfaceBuilder, because we aim to interoperate with ProjectBuilder and InterfaceBuilder on Rhapsody.

Dave: Have you firmed up plans for your sixth lane, PowerPlant?

Greg: What to do with PowerPlant is a very interesting question. Should we write a new version for Rhapsody? What should it be written in? Should we write it in Java? What about the MacOS version? A lot of questions, many unknowns. Here's what we've decided to do, given the dual operating system strategy at Apple and the current state of Java APIs.

One, we will continue to enhance PowerPlant for MacOS for the duration. Two, the transition of PowerPlant-based apps to Rhapsody will be achieved by linking with the Latitude library. We will achieve proof of concept of this by porting CodeWarrior, which is all PowerPlant code, using Latitude ourselves. Three, where Apple provides cross-platform APIs between MacOS and Rhapsody, support for these APIs will be rolled into PowerPlant where it is feasible and necessary.

Rhapsody already contains a very rich set of frameworks for desktop and Internet development. As such, it is not currently planned to build a new version of PowerPlant for Rhapsody, but rather to support Apple in its deployment of the Rhapsody APIs.

As far as doing a version of PowerPlant for Java, right now, we're talking about a pretty crowded field from AWT (JavaSoft), IFC (Netscape), AFC (Microsoft), JavaBeans (JavaSoft), and most likely a Rhapsody derivative from Apple. A conglomerate of Java APIs. Even if we did write a version of PowerPlant in Java, would it take hold? Would it be accepted as an industry standard as most of the systems vendors vie for leadership in the Java API space? The answer right now is no, so we're going to concentrate on the tools infrastructure for Java, but will not field a Java framework directly. We will, however, implement necessary Java class technology in order to support our Java RAD tools development. It's still too early to tell if it makes sense to make these Java classes available as a "framework". We'll have to wait and see.

Dave: Tell me about CW12. What's new in this release?

Greg: We'll have a final release of the 2.0 IDE (new project manager, multithreaded, multi-target, etc.) along with a pre-release of the 2.1 IDE which contains the integrated debugger (allowing users to set breakpoints directly from their source files).

We'll have a final release of the PPC JIT (Just In Time compiler), which gives greatly improved performance over previous versions, a beta release of the 68K JIT, a final release of the native Java compiler (i.e., not an interpreted Java compiler), and a release of the completely integrated VM containing the OBCI (Optimized Byte Code Interpreter) and JIT in a single binary. This is the same VM with which Microsoft's Internet Explorer ships, and will be our new redistributable VM version.

We'll ship a single Debugger application for C/C++/Java/Pascal targeting MacOS (68K and PPC) and x86 cross debugging.

ANSI C++ support will be enhanced (explicit, mutable and improved default template arguments). We'll ship a prerelease of our Objective-C compilers and linkers for PPC and x86, allowing users to start developing for OPENSTEP and Rhapsody today from MacOS. We'll ship a prerelease of IR optimizing 68K compiler, with greatly improved codegen (10-15% better ByteMark scores) as well as a prerelease of improved MSL C++ libraries, which along with the new prerelease ANSI C++ compiler, generates much smaller executables and improves compile speed.

Dave: In the same vein, what's on tap for CW13?

Greg: We'll ship the release version of the 2.1 IDE, which contains the integrated debugger, new text engine, and portable project format (single project file will work on all supported platforms). There will be a prerelease of the 2.2 IDE (with contains a "sneak peak" at our integrated RAD support) and a prerelease of the native Rhapsody IDE, which will run as a full fledged Yellow Box application, allowing users to develop and debug Rhapsody applications from within Rhapsody itself.

We'll see continued Java VM/JIT performance improvements, as well as reducing size overhead. We'll add native Java codegen support allowing use of Java to target PowerPC native executables, giving greatly improved performance without the use of a JIT. We'll also ship a prerelease of a completely native Java compiler written in C++, allowing users to use Java to generate native executables for all Code Generators which are supported by Metrowerks (currently 68K, PPC, x86, and MIPS)

CW13 will see improved Java debugging support and two machine debugging, allowing users to debug between any two machines running MacOS, Rhapsody, and Windows.

We'll add name-spaces support in the ANSI C++ compiler, along with other C++ improvements and a prerelease of Direct to Objective-C support in the ANSI C++ compiler.

Finally, we'll ship both Constructor for Windows (allowing users to generate Windows resource files visually from MacOS) along with a prerelease of Constructor for MFC, which allows users to create their MFC views from within Constructor, and then have Constructor generate all the source needed to instantiate the views. MT

Here's a limited time offer that'll have you seeing double. Order the MacTech CD-ROM™ Volume 11 and get the upgrade (expected release 1Q97) absolutely free! We'll even throw in the shipping costs for the upgrade. And as if that wasn't enough, Volume 11 is discounted too – **now just $79!**

That's right, so for less than you would normally pay for one CD... you'll get two! So hurry, pick up the phone, fire up the e-mail, launch that fax machine, or simply drop by our Web site and score yourself this double-whammy of a deal!

MacTech CD-ROM Vols. 1-11

- 1420+ articles from all 127 issues of MacTech Magazine (1984-1995)
- Improved hypertext and new THINK™ Reference Viewer 2.2
- 100+ MBs of source code
- Full version of THINK Reference 2.0
- 80MBs of FrameWorks/SFA archives
- Sprocket™! MacTech's Tiny Framework that compiles and supports System 7.5 features
- The best threads from Macintosh programmer newsgroups, plus thousands of notes, tips, snippets, gotchas and much more!

Developer DEPOT

Web Site: http//www.devdepot.com • E-mail: orders@devdepot.com
Phone: 1-800-MACDEV-1 • Outside the U.S. & Canada: 805-494-9797 • Fax: 805-494-9798

By Jessica Courtney

Open Door Networks Ships HomeDoor 2.0

Major new upgrade to industry-pioneering product

Open Door Networks, Inc. announces the shipment of HomeDoor 2.0, a major new release of the product that started the Macintosh multi-domain Internet services market. HomeDoor 2.0, which works both conventionally and as a plug-in for WebSTAR and compatible servers, provides Webmasters with unprecedented levels of flexibility in setting up multi-domain Web servers using the Macintosh. HomeDoor remains the only Macintosh product to offer "any browser, any server" compatibility, and with the 2.0 release it becomes the only product to provide Webmasters with the option of implementing either or both of the two current mechanisms for providing multi-domain Web service.

HomeDoor 1.0 pioneered the multi-domain Internet services market on the Macintosh by providing Mac Webmasters with the ability to serve Web pages for multiple independent domains from a single Macintosh server. Apple Computer quickly recognized HomeDoor's value by bundling the product with its Apple Internet Server Solution soon after HomeDoor's introduction. Open Door continued to grow the multi-domain services market through the introduction of its LogDoor and MailDoor products, providing single server, real-time monitoring/logging and e-mail on a domain-by-domain basis. Most recently, Open Door introduced its Multi-domain Webmaster Suite, an integrated set of multi-domain products and accessories.

With the evolution of the World Wide Web and its underlying protocol, the HyperText Transfer Protocol (HTTP), a new technique for providing multi-domain Web service has recently become available. HTTP 1.1 includes a "host" field, which enables an alternate way of providing multi-domain Web service. The host field is sent by most current Web browsers, but is not supported by some old browsers, including Netscape Navigator version 1.1 and many America Online browsers. Nonetheless, in certain situations, "host field mapping" does provide advantages over the "any browser, any server" technique pioneered by HomeDoor 1.0 (see <http://www.opendoor.com/MultiDomainFAQ.html>). To provide for maximum flexibility, HomeDoor 2.0 offers the Webmaster a choice between techniques on a domain-by-domain basis, enabling Webmasters to "mix and match" as desired.

HomeDoor 2.0 also includes a completely redesigned Admin application. Written from the ground up as a full-fledged Macintosh app, HomeDoor Admin 2.0 integrates both multi-domain Web service techniques through a single, consistent user interface. HomeDoor 2.0 also incorporates new features, such as the ability to export configuration data to a tab-delimited text file.

All Open Door products, including HomeDoor 2.0 can be ordered securely from <http://www.opendoor.com/order.html>.

Apprentice 6 is Now Available!

Weighing in at well over 600 megabytes, you'll find hundreds of high-quality and up-to-date source code examples. And all of the source code is either new or updated for this release! Included are some of the coolest new code, designed specifically to demonstrate the latest techniques, and all of the code is in C/C++, or Pascal, using CodeWarrior, Symantec, and MPW. You'll find complete working examples of applications, games, control panels, extensions, utilities, and more.

If you are new to programming, the included "frameworks" will help tremendously. These are complete program shells that take care of many of the tedious tasks associated with Mac programming, including menus, standard dialogs, file handling, and the like.

Looking for libraries or routines designed around a specific task? Apprentice contains dozens of libraries and classes (including a complete suite of PowerPlant and Think Class Libraries), from graphics and sounds to menu management and TCP/IP communications. Many of the libraries include complete source code. If you are just looking for a small routine to do a specific thing, you'll find hundreds of useful code "snippets", small routines that perform specific programming tasks.

Programming languages interest you? There are over a dozen complete programming environments to suit any taste. Basic, Clean, Eiffel, Lisp, ML, Modula-2, Python, and Tcl-Tk are only a few of the languages included. Most come complete with documentation and programming examples, and some include complete source code to the language itself.
<http://www.celestin.com/upgrade.html>.

Roaster Technologies Licenses Object Design's PSE for Java to Integrate with Roaster Release 3

Roaster Technologies today announced that Roaster Release 3 will include Object Design's ObjectStore PSE for Java. Roaster, the pioneering Integrated Development Environment for Java on the Macintosh, has added significant functionality by tightly integrating ObjectStore PSE into its new release.

Written entirely in Java, ObjectStore PSE for Java is a high-speed, small-footprint database that provides portable, transparent Java object persistence. This integration will enable Roaster users to access data from their Java applications easily, without having to map Java objects to sequential files or tables. When combined with the ability of Roaster Release 3 to create stand-alone Java applications, the possibilities afforded Roaster owners by this integration are limitless.

Developers who use Roaster will have the added bonus that Netscape has licensed PSE for Java for inclusion in their upcoming Communicator. Therefore, Roaster developers will be able to write applets that allow users of Communicator to leverage the distributed computing environment of the Web.

In addition to licensing ObjectStore PSE for Java, Roaster Technologies will port Object Design's ObjectStore PSE Pro for Java to the Macintosh and resell it to Roaster customers. ObjectStore PSE Pro for Java adds advanced transaction-level recovery and support for multiple databases from a single application. Roaster's edition of ObjectStore PSE Pro for Java will be available soon for electronic purchase from a secure commerce server available on the Roaster Website at <http://www.roaster.com>.

Leverage the Power of your AppleScript and 4th Dimension Applications with ScriptAgent.

ScriptAgent embeds AppleScript into the 4D procedural language, which enables 4D to communicate bi-directionally with virtually any AppleScriptable application.

A user can enter a customer code into a 4D layout, press Tab, and instantly return the matching customer record into the layout from an outside source (like MacP&L, an AppleScriptable accounting application). Or build a complete publishing workflow solution using QuarkXpress, 4D, and your favorite scriptable graphics cataloging application.

ScriptAgent works great with the new 4D Version 6, and will provide the developer with massive background processing power using the new EXECUTE ON SERVER command.

ScriptAgent includes a complete users guide and tutorial in Acrobat format, tutorial sample application, and the AppleScript external for 4D. Upgrades are free of charge, as is Internet technical support.
<http://www.prodenhance.com>.

NeoLogic Augments Capabilities and Accessibility of Object Database Technology

NeoAccess 5.0 brings added performance, capabilities and full support for the latest releases of all major compilers and development frameworks

NeoLogic Systems, Inc., a leading provider of object-oriented database technology, announced the release of NeoAccess 5.0, a significant enhancement to the company's cross-platform, object database management system. Featuring added support for large databases (over four gigabytes) and expanded schema evolution support, NeoAccess 5.0 simplifies the development of sophisticated applications for a broad range of professional developers. NeoAccess 5.0 includes full support for the latest releases of all major compilers and application frameworks. It ships with full C++ source code to provide maximum flexibility for both commercial and corporate developers. NeoAccess 5.0 is available today for Windows, Macintosh and UNIX platforms.

NeoAccess reduces development time and costs for commercial and corporate application developers by offering fast storage and management of database objects in an easy-to-use class library.

NeoAccess supports all major compilers and application frameworks including Borland's OWL, Microsoft's Foundation Classes and MetroWerks' Powerplant.

Applications built with NeoAccess take advantage of its powerful caching system which keeps objects in memory, even after being disposed of by the application.

NeoAccess has been used as the backend database infrastructure for such applications as ClickWorks by Scitex Corporation, NetObjects' Fusion by NetObjects and the mail and newsgroup components in Communicator by Netscape.

A free evaluation copy of NeoAccess 5.0 sample applications and source code can be obtained via the NeoLogic corporate web site at <http://www.neologic.com>.

VOODOO 1.8 Integrates Version Control into CodeWarrior

New release 1.8 of the popular version control tool VOODOO offers scriptability and integration into Metrowerks' CodeWarrior IDE.

UNI SOFTWARE PLUS today announced that the new version 1.8 of its popular version control tool VOODOO will be released soon. In order to bring the new features to the customers as soon as possible UNI SOFTWARE PLUS further announced the immediate availability of a full functional free pre-release of VOODOO 1.8.

In version 1.8 the delta generation process has been improved again and is much faster. Besides that, the new version supports even the comparison of files of different types.

Right now VOODOO can display the differences of text files, MS Word files and resource files using different helper applications. Other file types will follow.
<http://www.unisoft.co.at/e/products/voodoo.html>.

VOODOO Engine For OEMs

In addition to the full version control tool UNI SOFTWARE PLUS will soon release a "VOODOO Engine" which is a server application with an AppleEvent interface that can be used to integrate version control functionality in other applications.
<http://www.unisoft.co.at/e/products/voodoo.html>.

PreFab's Player 1.1 — Essential Scripting Utility Gets Even Better

PreFab Software, Inc. announced that version 1.1 of Player is now shipping. Player adds verbs to AppleScript and Frontier to query and control non-scriptable applications and control panels. The 1.1 release offers new verbs, enhanced options, revised documentation and new example scripts.

PreFab Player 1.1 includes a feature for automating Adobe Photoshop. It is the only product on the market that can control Photoshop dialog items by name. Scripts can now "check" or "uncheck" a checkbox and call Player's standard query verbs, e.g. to verify that a certain item is enabled or execute a difference sequence depending on which radio button is selected. Photoshop 4.0 takes a preliminary step towards automation with built-in "Actions." Player is still required for the many real-world tasks that require the flexibility of scripting. Over the coming weeks, PreFab will highlight examples and customer success stories on its new website at <http://www.prefab.com>. MT

TIPS & TIDBITS

By Steve Sisak

File Manager Tip

If you've got a program, such as a compiler, which creates a temporary file, writes and reads from it, and then no longer needs the data, don't just close it and delete it; doing so usually causes the temporary data to be written to disk. Before closing it, call SetEOF(fRefNum, 0); This tells the disk cache that none of the data associated with the file need ever be written.

Jorg Brown
<brown@connectix.com>

Running m68k Codes "Compiled-in" (embedded) in a PowerPC Program

Here is a tip I bumped into recently. I wanted to run a sequence of m68k instructions (actually, to perform a system trap) on a PowerMac. Usually, this is a trivial task: compile the code in the 68k universe (for example, with in-line assembly supported by many m68k C/C++ compilers). From within a PowerPC application, one can call PowerPC glue code to the trap, which is probably present in the 'InterfaceLib'.

Unfortunately, neither of these easy options would apply in my case: I only have a PowerPC compiler (I don't have disk space to install both 68k and PowerPC parts of the CodeWarrior). The trap I wanted to call (_ReadXPRam), is not an "official" one, thus there is no glue code for it in PowerPC libraries. Nevertheless, I found an easy way to run a sequence of 68k codes "compiled in" a PowerPC executable, and pass data in and out. By the way, this is working code that prints the contents of the xPRAM (more on xPRAM can be found at <http://pobox.com/~oleg/ftp/xPRAM.html>).

```
// Printing out the contents of the xPRAM
// The trick is that _ReadXPRam/_WriteXPRam traps are available only
// from within 68K universe. So, if this code runs in the PowerPC mode,
// we've got to switch universes before running M68K code sequences...

#include <stdio.h>
#include <MixedMode.h>

    //This is
    // CLR.L  D0        address would be 0
    // MOVE.W  $4(A7),D0   size -> lo word of d0
    // SWAP  D0         size -> hi word of d0
    // MOVEA.L     6(A7),A0   where -> A0
    // _ReadXPRam
    // MOVEA.L (A7)+,A0    standard PASCAL epilogue
    // ADDQ.W#$6,A7
    // JMP  (A0)
    // RTS
    //This is a sequence of M68K instructions; unfortunately,
    // a PowerMac compiler doesn't understand them. So we've
    // got to assemble by hand <sigh>
    //
    // BTW, to write into xPRAM, replace 0xA051 in the sequence
    // below with 0xA052
    //pascal void read_extended_PRAM(char * where, const short size) =

static short read_extended_PRAM []=
{ 0x4280, 0x302F, 0x0004, 0x206F, 0x006, 0x4840, 0xA051,
0x205F, 0x5C4F, 0x4ED0  };

#define COMP_NORET_2(name, a1, a2)                            \
  name##_procinfo = kPascalStackBased                         \
     | STACK_ROUTINE_PARAMETER(1, SIZE_CODE(sizeof(a1)))      \
     | STACK_ROUTINE_PARAMETER(2, SIZE_CODE(sizeof(a2)))

#define RD_ALLOC(routine) static RoutineDescriptor          \
  routine##_RD = BUILD_ROUTINE_DESCRIPTOR                    \
  (routine##_procinfo, routine)

enum {
  COMP_NORET_2(read_extended_PRAM,char *,const short)
};

void main(void)
{
  unsigned char whole_xPRAM_buffer[256];
  UniversalProcPtr u_read_extended_PRAM =
  NewRoutineDescriptor((long (*)())read_extended_PRAM,
            read_extended_PRAM_procinfo,
            kM68kISA);
  CallUniversalProc(u_read_extended_PRAM,
            read_extended_PRAM_procinfo,
            whole_xPRAM_buffer,
            sizeof(whole_xPRAM_buffer));
  DisposeRoutineDescriptor(u_read_extended_PRAM);

  printf("\ncontents of the xPRAM\n");
  for(register int i=0; i<sizeof(whole_xPRAM_buffer); i+=16)
  {
    printf("\n%04x  ",i);
    for(register int j=0; j<16; j++)
      printf("%s%02x", j%4 == 0 ? " " : "",
        whole_xPRAM_buffer[i+j]);
  }
}
```

Oleg Kiselyov
<oleg@pobox.com>

MT

*Send us your tips or we'll install EvenBetterBusError on your machine! On the other hand, we might just pay you **$25** for each tip we use, or **$50** for Tip of the Month. You can take your award in goods, subscriptions or US$. Make sure any code compiles, and send tips (and where to mail your winnings) to our **Tips e-mail address** at tips@mactech.com. (See page 2 for our other addresses.)*

MacTech™ Now!

http://www.mactech.com

(...and we really mean now.)

Think of it as your source of news and announcements in the MacOS developer community. Loaded with up to the latest news, and constantly updated with developments in our industry.

MacTech NOW™ brings you up to speed on everything you need to know – instantly! And thanks to our new "fast-download" design, you'll get to the information you want in seconds.

Give it a spin! Check it out today and get access to over 1500 pages loaded with news, tips, programming secrets, product reviews, and much more. And for those of you looking for some kicks, there's the ever popular "Programmer's Challenge" section where you'll get to bang heads with the best in the community.

Log on to MacTech NOW. Things are happening right now, and you should be aware of them.

http://www.mactech.com

ADVERTISER/ PRODUCT LIST

LIST OF ADVERTISERS

A.D. Software.	27
Adianta Inc.	41
Adobe Systems Incorporated	26
The AG Group	51
Aladdin Knowledge Systems Ltd.	5
Aladdin Systems, Inc.	20-21
Altura Software	28
Apple Developer University	37
Bare Bones Software, Inc.	65
BeeHive Technologies, Inc.	7
Biit, Inc.	31
Bowers Development	43
Conix Graphics	14
Crawford Software Consortium	46
Datapak Software Inc.	13
DC Micro Development	47
Developer Depot	111
dtF Americas, Inc.	35
Excel Software	33
Faircom Corporation	9
FlexWare International LLC	68
Griffin Technologies	11
JointSolutions Marketing	67
Late Night Software.	30
MacHack Conference	60
MacTech CD ROM	73
MacTech™ Now!	77
MacXperts.	49
Main Event	39
Mango Tree Software, Inc.	62
Mathemaesthetics, Inc.	1
Metrowerks	BC
Micro Macro Technologies, Ltd.	15
MindVision Software	45
Miracle Software, Inc.	36
Motorola	IBC
Neologic Systems	29
Onyx Technology	19
Perceptive Scientific Instruments	49
Perforce Software	40
PreFab Software, Inc.	17
PrimeTime Freeware	38
Quasar Knowledge Systems	18
Ray Sauers Associates	63
Royal Software, Inc.	59
Scientific Placement	49
Seapine Software, Inc.	32
SmartCode Software, Inc.	25
StepUp Software	54
Stone Tablet Publishing	55
Symantec	IFC
The Trattner Network	49
UNI SOFTWARE PLUS	17
Water's Edge Software	52

LIST OF PRODUCTS

ADB I/O • Beehive Technologies, Inc.	7
Adobe Developer Tools • Adobe Systems Incorporated	26
AppMaker • Bowers Development	43
BBEdit • Bare Bones Software, Inc.	65
C/C++ SDK• Motorola	IBC
CodeWarrior™ • Metrowerks	BC
Crusher!™• DC Micro Development	47
Developer Tools • Developer Depot	79
DragInstall • Ray Sauers Associates	63
dtF, The Relational Database System • dtF Americas, Inc.	35
d-tree™ • Faircom Corporation	9
EtherPeck™ • The AG Group, Inc.	51
FlexWare® • FlexWare International LLC	68
Guide Composer • StepUp Software	54
Heap Manager • Biit, Inc.	31
InstallerMaker 4.0 • Aladdin Systems, Inc.	20-21
Installer VISE • MindVision Software	45
MacAnalyst • Excel Software	33
MacDesigner • Excel Software	33
MacHack Conference • MacHack	60
MacHASP • Aladdin Knowledge Systems Ltd.	5
MacRegistry™ • Scientific Placement	49
The Memory Mine™ • Adianta Inc.	41
MicroGuard Plus™ • Micro Macro Technologies, Ltd.	15
MK Linux •PrimeTime Freeware	38
neoAccess™ • Neologic Systems	29
Object Master™• Altura Software	28
ObjectSet Mail SDK • SmartCode Software, Inc.	25
ODBC Driver • Faircom Corporation	9
OO File™• A.D. Software	27
OpenPaige™• DataPak Software, Inc.	13
Open GL for the Mac• Conix Graphics	14
Perforce • Perforce Software	40
Player Text Machine • PreFab Software	17
QuickTime VR Training • Apple Developer University	37
Recruitment • MacXperts	49
Recruitment • Perceptive Scientific Instruments	49
Recruitment • Scientific Placement	49
Recruitment • The Trattner Network	49
Resorcerer ® 1.2 • Mathemaesthetics, Inc.	1
r-tree® • Faircom Corporation	9
Scripter • Main Event	39
Script Debugger• Late Night Software, Ltd.	30
SiteWeaver• Miracle Software, Inc.	36
SmallTalk Agents® • Quasar Knowledge Systems	18
Spotlight™ • Onyx Technology	19
Stone Table ™ • Stone Tablet Publishing	55
StuffIt Installer Maker 3.1 • Aladdin Systems	20-21
TestTrack™ • Seapine Software, Inc.	32
TCP/IP Scripting Addition • Mango Tree Software	62
Tools for HyperCard, SuperCard, OMO & AppleScript • Royal Software, Inc.	59
Tools Plus™ 3.0 • Water's Edge Software	52
Visual Cafe™ for Java• Symantec	IFC
Voodo Locking Components • UNI SOFTWARE PLUS	17
Web Pages• Joint Solutions	67
WIBU-KEY • Griffin Technologies	11
Word Services Suite • Crawford Software Consortium	46

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

develop

Dear Apple developers,

You will be happy to know that Apple Computer and *MacTech* magazine have once again joined forces to ensure that you get the technical information you need to successfully develop for Apple platforms and to take advantage of Apple's market-leading technologies.

Starting with this issue of *MacTech* magazine, Apple's award-winning *develop* Technical Journal will be featured as a special section of *MacTech.* This means that develop will now be available to more developers than ever before.

Long considered the definitive source for insight, code, and vital programming information on Apple platforms and technologies, *develop* will continue to be reviewed by Apple engineers to ensure that it retains its accuracy and robustness.

And now, by being included in *MacTech*, articles will be more frequent and timely than ever before. In addition, *develop* readers will gain all of the benefits that *MacTech* has provided since 1984 – third party insights, advertising, product reviews, NewsBits and more.

Combined, *develop* and *MacTech* will provide you with even more of the high-quality technical information you have come to expect from each publication – all in one location.

So, whether you're a former *develop* subscriber or new to *develop,* we welcome you to *MacTech's* newest section, and we assure you that Apple and *MacTech* will continue to work together to provide Apple developers with information to enhance your programming efforts and increase your productivity!

David Krathwohl
Vice President
Developer Relations
Apple Computer, Inc.

Neil Ticktin
Publisher
MacTech Magazine

by Edward Voas

Appearance: Not Just Another Pretty Interface

Edward Voas

The Macintosh was once the model of consistency: every application behaved and looked the same, making the user feel at home. But that consistency faded as, due to lack of support from the Macintosh Toolbox, developers created custom controls, menus, and windows, moving forward (or sideways) with user interface innovations while the MacOS lagged behind. Now the Appearance extension takes the first step toward regaining that consistent look and feel we all remember so fondly, paving the way for switchable interface themes and making it easier to develop applications for the MacOS.

You've got this great idea for a user interface, but it means you have to write a slider CDEF. So you plug away, working to replicate what you've seen in dozens of applications, while dreaming of sliders that are available as part of the system. Well, your dream has come true. Meet Appearance, the biggest advancement of the Macintosh user experience since System 7. The Appearance extension provides sliders plus a lot more:

- Appearance implements a new look — Apple Grayscale — which was originally slated to be the default look of Copland, the former MacOS 8 plan. Under Appearance the standard system windows, controls, and menus all have the Apple Grayscale look.
- Appearance adds new controls such as progress bars, tabs, disclosure triangles, and sliders to the standard set, eliminating the need for developers to roll their own.
- Appearance extends the Window, Control, Dialog, and Menu Managers to provide functionality that's necessary for some of the new features to work correctly. Some of the new functionality fills in the gaps that developers have had to fill in on their own because the Macintosh Toolbox didn't support some necessary or desirable features. For example, the system MDEF now supports extended keyboard modifiers for menu item keyboard equivalents.

With Appearance you benefit the most by using as many of the system-supplied user interface elements as possible. Your user interface won't have a patchwork look — with the system elements, all pieces of the UI blend together nicely. Plus, as Apple enhances the UI elements, your applications can immediately (and automatically) benefit as new system versions are released. This becomes particularly important as we move toward switchable themes (explained below). Another major benefit is that your applications can be smaller, because you don't need to implement UI elements that are now supplied by the Toolbox. A sample application that demonstrates Appearance can be found at <http://www.mactech.com>.

Apple intends to supply an Appearance extention to be bundled with applications for users to install on MacOS 7.6 or earlier (later systems will contain the functionality of the SDK as part of the base system). Your application can determine whether Appearance is running by checking a Gestalt selector (gestaltAppearanceAttr). This selector returns a bitfield indicating which features of Appearance are in effect.

Edward Voas (voas@apple.com) is a staunch supporter of Truth, Justice, and Switchable Themes. He is currently the Technical Lead on Appearance and is the co-author of the popular shareware program Aaron. When Ed is not busy coding, he is hard at work memorizing lines from the Star Trek movies and boring his coworkers with inane facts from those movies ("What's the prefix code of the U.S.S. Reliant?").

THE ON-RAMP TO THEMES

You may have heard about themes at Apple's Worldwide Developers Conference (WWDC) or read about them in discussions on Copland. Essentially, a theme is an interface look that spans all elements of the user interface and ties them together with a certain graphic design. Themes are data driven — all the data that describes the theme interface is contained in a theme file. The data-driven aspect makes it easy to switch themes on the fly. **Figure 1** gives examples of three themes that were shown at the 1996 WWDC and elsewhere: Apple Grayscale, Gizmo, and High Tech.

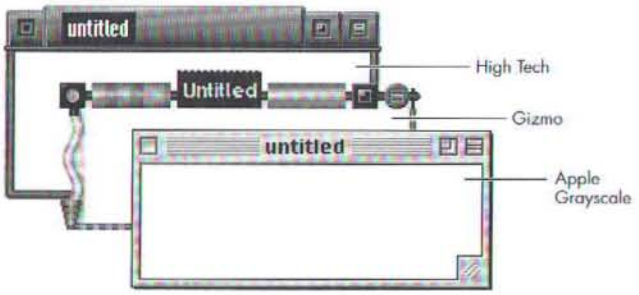


***Figure 1.** Three themes under Appearance.*

Now before you get too excited, please note that the theme-switching mechanism isn't implemented in the first version of Appearance; however, switchable themes are very much a part of the future of the MacOS. Appearance is the first step toward that future, and using the system controls, windows, menus, and other features provided by Appearance will allow your application to handle theme switches automatically when the time comes. I'll be referring to themes throughout this article, especially when talking about the Appearance Manager, which lets you get colors and patterns for the current theme.

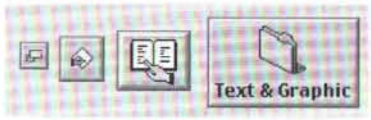
WHAT'S NEW AND IMPROVED

Appearance redesigns some old controls and provides many new ones. Windows sport a new look and added features, and there's a new help icon. Here we'll check out these snazzy Apple Grayscale UI elements and learn a bit about the new features. Then we'll look at the Appearance Settings control panel, which lets the user control theme variations and the system font, among other things. In later sections, we'll describe how the new features work and what you need to do to adapt your applications for Appearance.

NEW CONTROLS

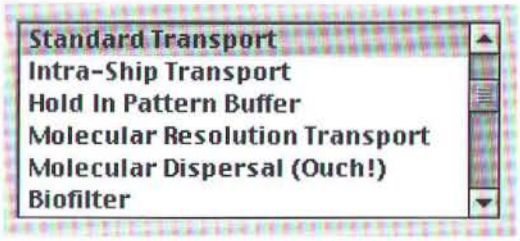
Many new controls are added to the system with Appearance.

Bevel button control



The bevel button control implements a rectangular button with a beveled edge. Typically, a bevel button displays an icon, but it can display an icon, a picture, or text, singly or in combination. You can also attach a menu to this control. Multiple bevel thicknesses and several different button behaviors are supported, to suit just about any use. This versatility makes bevel buttons well suited for use in toolbars or tool palettes. (These buttons should never be used to replace push buttons, however.) The sample application accompanying this article shows many different variations of bevel buttons; you'll be astounded by the possibilities.

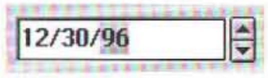
List box control



The list box control implements a simple list box. It requires an auxiliary resource of type 'ldes' to specify the features of the list, such as the number of columns and rows. (The Appearance Settings control panel, shown later, provides an example of columns in a list box.) This control allows filtering of keyboard events, and handles the default keyboard navigation you'd expect from a list box.

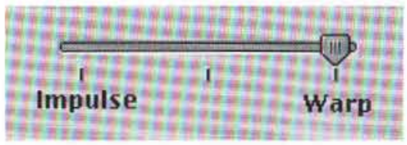
Note that if you use the list box control in a dialog, it won't respond to keyboard events — you can't tab into it, for example — unless the dialog has established an embedding hierarchy (described later). There's a similar restriction on the clock and editable text controls.

Clock control



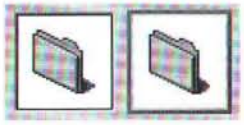
The clock control provides an editable date or time field, as you'd find in the Date & Time control panel. The little arrows next to the clock are part of the clock control. (Remember, you won't be able to tab or type characters into the clock control unless the dialog has established an embedding hierarchy.) You can also specify a noneditable version that simply shows the date or time. The noneditable clock permits live updating, so you can put a clock in your interface and let it tick away. The clock control uses the date and time formats set in the Date & Time control panel.

Slider control



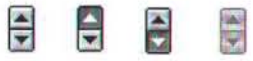
Sliders are finally part of the MacOS repertoire. You can choose which way the indicator faces or use a non-directional indicator. You can also specify whether to draw tick marks. The slider supports live feedback (described later).

Image well control



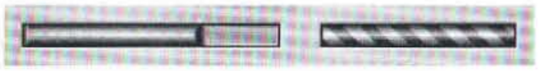
This control holds an icon or picture. In Apple Grayscale, image wells look like an editable text box with a picture inside instead of text. Image wells have a normal and selected state, as shown above. A good use for an image well is as a drop target for images or document icons. Drag and drop support isn't built into this version of Appearance.

Little arrows control



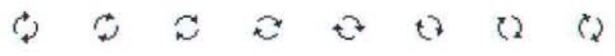
The little arrows control implements the little up and down arrows you often see tied to a box displaying a value. The arrows are used to increase or decrease the value. In the Memory control panel, for example, you click them to set the cache size or virtual memory partition size.

Progress bar control



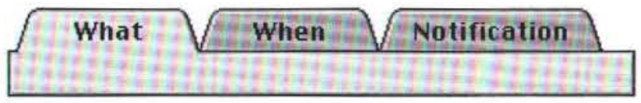
Progress bars are now part of the standard control set. You can tell a progress bar to switch into indeterminate mode, in which it displays an animated barber pole–like bar; you might use this mode to indicate that you haven't made a connection yet or are waiting for some piece of data before continuing. Because the indeterminate flag is separate from the value, you can switch back and forth without affecting the value.

Chasing arrows control



These are the spinning arrows that usually indicate an asynchronous process. In other words, there's something going on in the background but you can continue to work. You've no doubt seen them in Find File when searching for files.

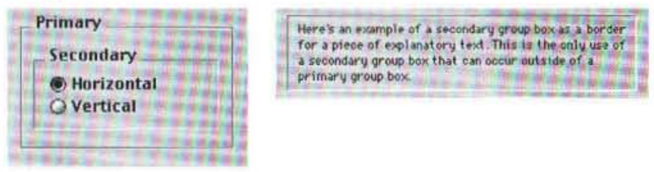
Tab control



Currently, the new tab control supports only one row of tabs running along the top of the control. Future versions will support more variants. As with the list box, you use an auxiliary resource to specify the tab names and any icons that appear beside the names.

You should try to restrict your use of tabs to those times when they're really necessary. Too many tabs can result in a very complex and confusing UI. Also, tabs can be difficult to localize for different script systems: the width of a text string can increase up to 50%, causing problems if you've set up the tabs to fit perfectly in a Roman script system.

Group box control

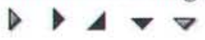


Here's an example of a secondary group box as a border for a piece of explanatory text. This is the only use of a secondary group box that can occur outside of a primary group box.

The group box control implements two group box looks — primary and secondary, as shown above. It also provides three different types of titles for the grouped items — text, checkbox, or pop-up menu. The pop-up title is useful for paged interfaces, such as the Sound and Speech control panels. You can embed other controls within the group box control, such as radio buttons.

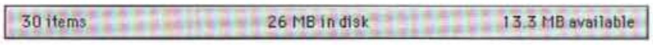
Primary groups should be used as the first level for grouping items. Secondary groups should always be inside primary group boxes. The only exception to this is if the group box is being used for a border around some text, as shown above.

Disclosure triangle control



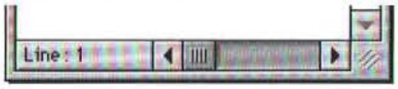
You can use the disclosure triangle control in places where you want to hide some information to reduce clutter but still give the user a way to view it. The user clicks the disclosure triangle to expose the hidden information. The Finder uses disclosure triangles next to folder icons when in list view.

Window header control



The window header control is what the Finder uses to draw the header of a window where text such as the disk information is shown. Text sold separately.

Placard control



The placard control implements a small panel like those often found at the bottom of a window to the left of the horizontal scroll bar. You might have seen them in CodeWarrior. You could use a placard control to show information such as the current line number, as shown above. In Apple Grayscale, placards look the same as the window header control, but this won't be the case in all themes, so be sure to use the appropriate control.

Visual separator control

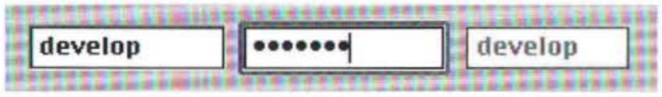
The visual separator control implements a simple divider line.

Controls to Replace Dialog Primitives

Dialogs can contain many items that are not controls — pictures, icons, editable text, static text, and user items. I'll be referring to such items as dialog primitives. Some of these dialog primitives now have control counterparts. For example, the counterpart of the editable text primitive is the editable text control, which is a control with all the functionality of the editable text primitive. In referring to items in dialogs, the word control refers only to items that are not primitives. Using the new controls allows you to take advantage of features provided by Appearance's embedding hierarchy.

If a dialog has an embedding hierarchy, the controls discussed in this section automatically stand in for their primitive counterparts. So the easiest way to use the controls is to let the Dialog Manager convert your primitives for you. You don't have to change your items into resource-based controls and create 'CNTL' resources to use these controls (though you could if you had your heart set on it).

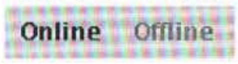
Editable text control



The editable text control replaces the old editable text dialog item. Because it's a control, it can be enabled and disabled. If you've ever tried to disable an editable text field in a dialog, you know how difficult this can be. With Appearance it's one line of code. The editable text control allows keyboard-event filtering and supports password entry. (Like the list box and clock controls, if you use the editable text control in a dialog, it won't respond to keyboard events unless the dialog has established an embedding hierarchy.)

The text in the control is always displayed on a white background in the Apple Grayscale theme. The figure above shows three editable text controls, one using the password entry variant. The control accepting passwords has the current keyboard focus, as indicated by the ring around it. (Keyboard focus is described later.)

Static text control



Static text controls replace the old static text items for embedding static text in dialogs. Since they're controls, they can be deactivated and will then be drawn disabled like other deactivated user interface elements. If you have a dialog with static text items in it and establish an embedding hierarchy for that dialog, the static text primitives will automatically become static text controls.

User pane control

The user pane control replaces the old user item construct in dialogs. This control is essentially a stub control that calls user-installed procedures to do its drawing, hit testing, and other things that controls do. It also lets you track the mouse and draw with the correct highlighting — normal, highlighted, or whatever. Even in its most basic form, the user pane is very useful because you can embed controls within it, which allows you to group items. When you hide, show, enable, or disable the user pane, the group of embedded controls will follow suit automatically.

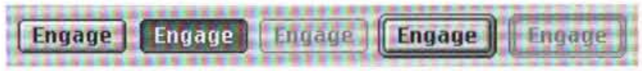
Icon control and picture control

These controls were created to stand in for icon and picture primitives in dialogs that have an embedding hierarchy. The icon control allows you to display icon suites, as well as the usual 'ICON' and 'cicn' icon types. When used to simply replace the old icon or picture primitive in dialogs, these controls don't track the mouse (they behave as icons and pictures always did in dialogs). You can create an icon or picture control and add it to a dialog through a 'CNTL' resource if you'd like it to track the mouse.

Changes To Old Controls

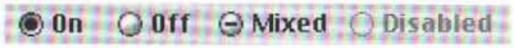
Some old controls have a new look, more features, or both.

Push button control



Button controls — now called push button controls to distinguish them from bevel button and pop-up button controls — have been changed to allow you to set a "default button" flag. When set, this flag tells the control to automatically draw the default ring (border) around the push button. The default ring is drawn outside the control's bounding rectangle. The default push button has a new look, as shown in the last two examples above.

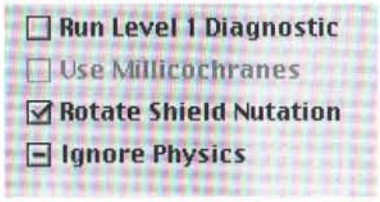
Radio button control



Radio buttons have a new look. They also support a mixed state. A mixed state is typically used when a radio button represents a selection that includes more than one state. Let's say you have a group of radio buttons that represent different planet classes: Class M, Class P, and Class K. As a user clicks a planet in a list, the radio buttons reflect the class of the selected planet. Now suppose the user selects four planets from the list box at once — for example, one Class K and three Class M planets. You can reflect this situation by putting the Class K and Class M radio buttons in a mixed state, indicating that some of the selection are of this type, but not all.

The kControlRadioButtonMixedValue constant is for setting the mixed value of a radio button. You can use it in a call to SetControlValue to set a radio button to the mixed state. Your radio button must have a maximum control value of 2 for this to work, since that's the value of the constant.

Checkbox control

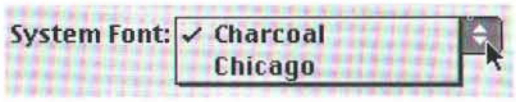


Our old friend the checkbox sports a new look and real checkmarks. The "X" variant of this control is still available for use in countries outside the U.S. Like radio buttons, checkboxes support a mixed state, as shown in the Ignore Physics checkbox above. The constant for the checkbox mixed value should look vaguely familiar — it's kControlCheckBoxMixedValue.

Scroll bar control

Scroll bars have a new look, too. One scroll bar variant supports live scrolling.

Pop-up button control



The pop-up button control is the old pop-up menu control with a new look.

The new Help icon

There's now a system-supplied help icon. You can combine this with a bevel button to get the new standard help-button look. The StandardAlert routine (described later) uses this approach to display its help button.

CHANGES TO WINDOW APPEARANCE

Appearance gives windows a new look, as shown in **Figure 2**. (Note that floating windows are now called utility windows.)

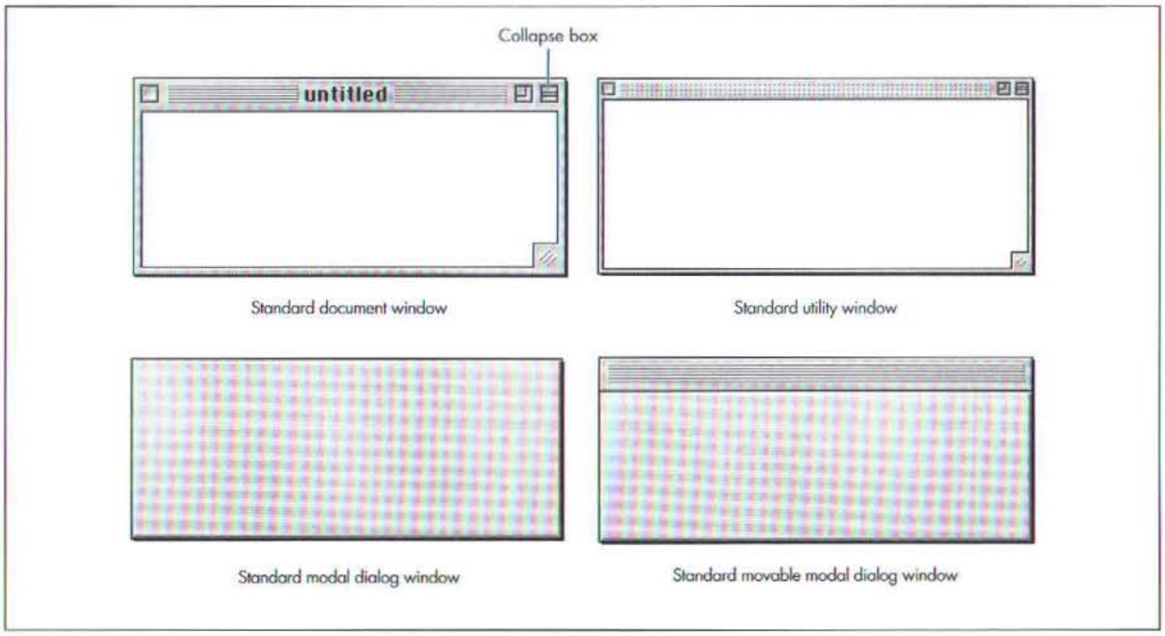
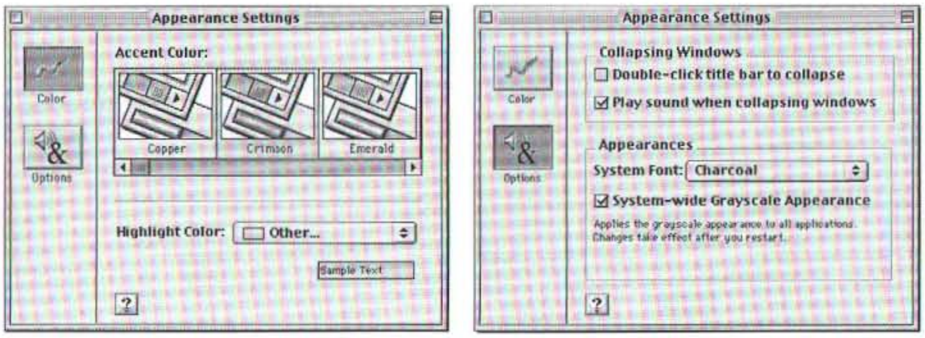


Figure 2. *Standard Appearance windows.*



Highlight Color: Other...

Collapsing Windows

☐ Double-click title bar to collapse

☑ Play sound when collapsing windows

Appearances

System Font: Charcoal

☑ System-wide Grayscale Appearance

Applies the grayscale appearance to all applications. Changes take effect after you restart.

Figure 3. *The Appearance Settings control panel*

The standard document window has a thicker border than it used to. This border is functional: it allows you to grab the window and drag it from any side. Alerts are distinguished from dialogs by a red border; the new movable alerts also have a reddish title bar (giving new meaning to the term *red alert*). Remember to use alerts only to warn the user of something or to present important information; in all other cases, use a dialog. The many window variants are described later.

Document windows (and utility windows) also have a new element: the collapse box. Clicking it collapses the window into a title bar. You may wonder why the new collapse box is where the zoom box used to be and the zoom box has moved over. For consistency, the most frequently used widget is always on the outside — in this case, the collapse box. (More windows have collapse boxes than have zoom boxes, and no window can have a zoom box without a collapse box.)

Also, the size box is now integrated into the frame of the window. If you're adapting these new windows directly (as described later), the size box is drawn for you automatically. You don't need to call DrawGrowIcon.

The Appearance Settings Control Panel

The new Appearance Settings control panel (**Figure 3**) allows the user to control these settings:

- The accent color of the current theme. This affects the coloring of menu items as they're chosen, scroll bar and slider indicators, progress bar indicators, and focus rings.
- The highlight color. This affects the coloring of any highlighted item — for example, selected text.
- Window collapsing. When Appearance is running, the WindowShade control panel is removed. When "Double-click title bar to collapse" is checked, the user can double-click the title bar in addition to using the collapse box to collapse a window.
- The system font. The Apple Grayscale system font is Charcoal, but users can choose the Chicago font if they want.
- System-wide Grayscale Appearance mode. When "System-wide Grayscale Appearance" is checked, it means that every application gets the Apple Grayscale look; otherwise, only applications that have explicitly adopted Appearance's features have the look.

Not Your Father's Control Manager

To make the new controls work correctly, it was necessary to add some features to the Control Manager. The Control Manager now does the following:

- Supports embedding, to make drawing order and hit testing predictable and enable some Appearance features.
- Supports keyboard focus for controls.
- Makes it easy to get and set control data and choose fonts for control titles.
- Supports live feedback for scroll bars and sliders.
- Provides a mechanism for controls to advertise the features that they support.

Drawing Order and Hit Testing

From the very beginning, the order in which controls draw in a window has been backwards due to how the Control Manager manages the control list for a window. As controls are created, they're added to the head of a window's control list. When the controls are drawn, the list is traversed, yielding a drawing order opposite to the order in which they were added to their owning window. To confuse things more, in normal dialogs, dialog primitives, such as editable text and static text items, are always drawn from first to last. When you have a mixture of controls and dialog primitives, the primitives are drawn from first to last after all the controls are drawn from last to first.

This isn't a problem if you can assume that controls don't overlap or contain other controls. With new controls such as tabs and group boxes, however, you can't make this assumption. Consider the case where you want a tab control containing three radio buttons and an editable text field. Let's say you add them to the 'DITL' resource in this order: first the tab, then the three radio buttons radio 1, radio 2, and radio 3, and then the editable text item. When they're drawn, you get this order: radio 3, radio 2, radio 1, tab, editable text. You've just covered up your radio buttons with the tab control (see **Figure 4**)! This happens because controls are drawn first, and then dialog primitives. Needless to say, trying to manage the drawing order can be difficult.

Hit testing has similar problems when items are inside tabs, group boxes, and other such controls. The FindControl routine uses a linear search over the control list to find the first control that returns a part code other than kControlNoPart. This isn't always accurate, as disabled controls are skipped even though they were hit. FindDialogItem also uses a linear search of the dialog items, and stops when it finds the first enabled item that a given point is in. Consider what happens with that tab control with the three radio buttons if your application calls FindDialogItem with a point that's in one of the radio buttons. FindDialogItem never finds the radio button because the first item searched is the tab control. The point is certainly within the tab control (the radio button is inside the tab), so FindDialogItem returns the dialog item number of the tab control. The right approach is to do an "inside-out" hit test to find the most deeply nested control hit by the mouse. Read on to see how Appearance makes this possible and simplifies managing the drawing order.

The Embedding Hierarchy

To make drawing order and hit testing predictable and easy to follow, we've added an embedding hierarchy, where controls can be embedded within other controls, giving you a rudimentary "view system." Embedding is a really exciting aspect of Appearance. The hierarchy ensures that parent items are always drawn before their children. It also helps hit testing since the hierarchy can be traversed quickly to find out which control the cursor is over. A hierarchy can exist in any window. It's not restricted to dialogs, though it's easiest to use there, since the Dialog Manager deals with focus management and event handling for you.

Root controls

To enable control embedding in a window, you must create a root control for that window. The root control is the container for all other window controls. You create the root control in one of two ways: by calling the CreateRootControl routine or by setting a dialog flag to tell the Dialog Manager to create one for you (more on this later). You can't embed controls without a root control, and any attempt to do so will result in all the money in your bank accounts being transferred into mine. So watch it.

When a window has a root control, calls to NewControl (and GetNewControl) automatically add controls to the root of the window. Calling EmbedControl or AutoEmbedControl is the only way to explicitly change this. You use EmbedControl to specifically embed one control in another. The Dialog Manager uses AutoEmbedControl when creating items from a 'DITL' resource.

AutoEmbedControl uses visual placement to automatically determine what control, if any, a control should be embedded within, based on bounding rectangles. For example, going back to the tab and radio buttons in **Figure 4**, the Dialog Manager would have embedded the radio buttons and editable text field in the tab control for two reasons: they came after the tab in the 'DITL' resource and they fit inside the tab control. The ordering of 'DITL' items is still important — a control can be embedded automatically only in a control that already exists — but the results are a lot more predictable with embedding. Create your elements from back to front and everything will

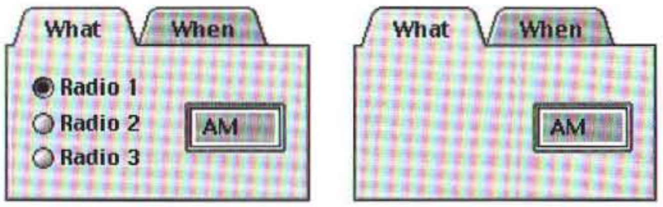


Figure 4. *The problem with control drawing order*

fall into place. The sample code accompanying this article provides some good examples of how to do this.

Groups and latency

The embedding model has many advantages, one of which is that it makes it possible to treat a set of items as a group. By acting on a common parent, you can move, disable, or hide groups of items. Disabling the root control of a window, for example, will disable all items in the window.

Doing things like switching tabs becomes remarkably easy. You can simply use a blank user pane control as the common parent for all items in a particular "page" of a tab control. After creating as many user panes as you have tabs, you can just hide one and show the next when a tab is clicked. All the controls embedded in the user pane will be hidden and shown automatically when the user pane is hidden and shown.

In hiding, showing, disabling, and enabling groups of controls, it's important to preserve the state of an item when it's hidden or disabled so that when its parent is shown or enabled, the item appears in that same state. To accomplish this, we've added the concept of latency. Controls are considered latent when they're disabled or hidden only because their parent control is disabled or hidden. It's effectively saying, "This item should be enabled (or shown) when its parent is enabled (or shown)." If you disable a control that's latent, it becomes truly disabled and won't be enabled when the parent is enabled. Likewise, if you enable a control whose parent is disabled or latent, the control becomes latent until its parent is enabled.

When enabling and disabling controls, to ensure that this latency information is always correct, you should use the new routines DeactivateControl and ActivateControl instead of just setting the highlight with HiliteControl, as you undoubtedly have always done in the past. It would be smart to use these routines even when no embedding or latency is involved — they'll set the highlight code correctly and redraw the control. For controlling visibility, the old HideControl and ShowControl routines have been modified to deal with latency when an embedding hierarchy is present for a window.

Focus Management

With Appearance, you can get and set the keyboard focus of a window. The control with the keyboard focus is the one that receives all keystrokes. For example, the Dialog Manager tests to see which control has the focus when a keyboard event is processed and sends the event to that control. It's possible for nothing to have the focus, in which case the keystroke is simply discarded. As mentioned earlier, to indicate that a particular control has the keyboard focus, a focus ring is drawn around the control. (It's a lavender ring by default, but the user can choose a different color in the Appearance Settings control panel.)

The keyboard focus routines introduced with the Appearance extension are available only when a root control has been created for a window. In windows with an embedding hierarchy, you can use several routines to get, set, advance, reverse, and clear the keyboard focus. The default focusing order is a simple linear progression through all the enabled, visible controls in a window. The order is based on the order in which controls are added to the window. This is the same approach as using the order in which editable text items appear in a DITL to control tabbing order. (Eventually, other system-supplied focusing heuristics may be available, such as spatial focusing that's based on the visual placement and grouping of controls, not on the order in which controls are added.)

Currently, the controls that support keyboard focus are the editable text, list box, and clock controls. In future versions of Appearance every control that can receive user input will support keyboard focus (push buttons will, but visual separators won't, for example). You can plan for that day by ordering your controls so that the focus will move from one to the next in the order you desire and by making sure they have enough space around them to allow for focus rings. Focus rings are outset a maximum of three pixels from the control's bounding rectangle.

Getting and Setting Control Data

Developers need to get and set different attributes of a control. In most cases, these attributes are unique to a particular control. In the past, the only way to allow access to control-specific information was to create a handle to hold such data, place it in the contrlData field, and publish the interface. A good example of this is the menu handle of a pop-up control. Unfortunately, this approach makes it hard to change the control implementation in future versions of the control.

In Appearance, we've added a mechanism by which controls can allow the outside world access to their specialized data without exposing how it's stored. This data access mechanism is the cornerstone of many of the new Appearance features, allowing you to get and set control fonts, user-pane callback functions, bevel-button image information, and other useful things. Two new CDEF messages implement the data access mechanism: kControlMsgGetData and kControlMsgSetData. To advertise that a CDEF supports these messages, it must return kControlSupportsDataAccess in its feature flags.

Each piece of information that a CDEF wants to provide access to is referenced by a tag. A tag is some constant that is meaningful to the CDEF and represents the data in question. For example, to get at the clear text of a password field (the actual password), you would use the tag kEditTextPasswordTag. To set the indeterminate flag for a progress bar, you would use kProgressBarIndeterminateTag.

Each tagged piece of data can be any data type. It might be a menu handle, a UniversalProcPtr, or a structure. It's up to the creator of the CDEF to define the tag and the data type for the data that's passed back and forth.

To get and set data, you use the GetControlData and SetControlData routines, which use a format similar to that used by the Collection Manager and Apple events. These calls are pointer based, and storage is always owned by the caller, which means that if you're trying to get the menu handle from a bevel button control, for example, you would pass in a pointer to a menu handle. To get a vital warp-engine matter/antimatter intermix ratio from a control (a Fixed value, for those who don't know), you might call GetControlData like this:

```
Fixed    theRatio;
OSErr    err;
Size   actualSize;

err = GetControlData(myControl, 0, kWarpIntermixTag,
sizeof(theRatio),
     (Ptr)&theRatio, &actualSize);
```

Better font control

Appearance allows you to set the font of any control title, independent of the system font or window font. Previously, your only choices were the system font and, if the control supported that variant (and most did), the window font. Now you can set the control title font to any font your heart desires.

There are also new constants available to take advantage of some system-defined fonts — big system font, small system font, and small emphasized system font. In the U.S. version of Apple Grayscale, those fonts are Charcoal 12, Geneva 10, and Geneva 10 bold, respectively. By using system-defined fonts, you can be sure to get the correct font when running with a different script system, such as Kanji. This helps your programs adapt automatically to different locales. Never, ever hard-code a font number or font size. I mean it!

To save you the hassle of setting control title fonts for each item after a dialog is created, we've added the new 'dftb' resource type for a dialog font table. The 'dftb' resource is used to specify the initial font settings for each item in a dialog. This resource runs parallel to a 'DITL' resource and has an entry for each item in a dialog. It should have the same resource ID as the 'DITL' resource for the dialog. Of course, you don't need to create a 'dftb' resource if you want to use the system or window font.

There's an old resource — the 'ictb' resource — that very few people seem to know about or use. The 'ictb' resource allows you to set the font information for editable text and static text items, but not controls. Since it's now possible (and desirable) to change the font of individual control titles, Appearance adds the new 'dftb' resource. The 'dftb' resource has a straightforward resource format, making it easier to create and maintain than the old 'ictb' resource. If a dialog with no embedding hierarchy has both a 'dftb' resource and an 'ictb' resource, the 'ictb' resource is used for the static text and editable text items only. If the dialog has an embedding hierarchy, any 'ictb' resource information is ignored.

Live scrolling

As mentioned earlier, Appearance allows you to have real live feedback with scroll bars and sliders. With some variants of the scroll bar and slider controls you can set an action procedure to be called back as the indicator is moved. Each time the action procedure is called back, the value of the control will indicate what position the user has dragged the indicator to. Listing 1 shows a good example of how to install and use a live feedback callback.

Be sure you set the action proc to be a ControlActionUPP, not the DragGrayRgnUPP that normal indicator dragging uses. If an action proc isn't passed into TrackControl or set with SetControlAction, no live scrolling occurs — the control reverts to dragging a ghost of the indicator.

The control features set

To get the set of features supported by a control, you can call GetControlFeatures. This new routine returns a bitfield in which each bit represents a feature the control supports, such as keyboard focus or data access. If you want to write a CDEF that supports any of the new features, the CDEF must respond to the kControlMsgGetFeatures message.

OLD DIALOG MANAGER, NEW TRICKS

The new MacOS user experience provided by Appearance calls for some new features to be added to the Dialog Manager. For example, you can now ask that a dialog's background fit the theme, that a root control be created for a dialog automatically when the dialog is created, or both. These features are requested through a bitfield where each bit represents some feature. This bitfield can be specified in two ways: through the new NewFeaturesDialog routine or in the new dialog and alert extension resources. In addition, the Dialog Manager provides an enhanced standard alert routine and lets you request standard movable modal behavior for dialogs and alerts.

New resource types for dialogs and alerts

To make it simpler and more straightforward to add the new flags, we've created two new resource types to hold extended information for dialogs and alerts: 'dlgx' and 'alrx'. These resources specify the feature bits and other information, such as a window title for movable alerts. You relate these resources to 'DLOG' and 'ALRT' resources by their ID. If you created a 'DLOG' resource with an ID of 128, you would create a 'dlgx' resource with an ID of 128 to add extended information. The Dialog Manager looks for a 'dlgx' or 'alrx' resource after reading in the 'DLOG' or 'ALRT' resource.

Those who want to create dialogs programmatically can use the NewFeaturesDialog routine. NewFeaturesDialog is identical to NewDialog (and NewColorDialog), except that it takes a flags parameter. These flags are the same ones that you would set in a 'dlgx' or 'alrx' resource.

Automatic control creation

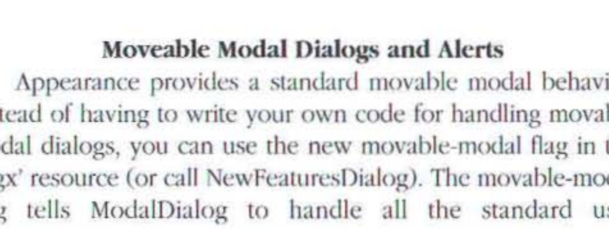
If you set the features flag for automatically creating a root control in a dialog, be aware that all dialog primitives are replaced with their control counterparts. The Dialog Manager routine GetDialogItem still works as it always did: if you call GetDialogItem on a static text item, you'll get back a handle to text, not a handle to the control. This call has been modified to sense when an embedding hierarchy is present and to talk to the controls to get the appropriate data. If you want to access the actual control for a static text item, you would use the GetDialogItemAsControl routine. You can then act on the item as you would with any control. The SetDialogItem routine works as it always did except for a few restrictions when the dialog has established an embedding hierarchy. In that case, you can't change the type or handle of a dialog item (except for user items, for which you can still set the drawing procedure).

Converting dialog items into controls makes it possible to do things you couldn't do before. For example, when all dialog items are controls, you can highlight, enable, and disable everything in a dialog, including static and editable text items. Now it's as simple as a call to DeactivateControl or ActivateControl.

In fact, when an embedding hierarchy is established in a dialog, if you deactivate the dialog, all dialog controls are automatically disabled. When the dialog is reactivated, the items are reactivated. This is the desired behavior and part of the new Human Interface Guidelines. Only the frontmost window should have active controls and other UI elements. This helps distinguish the active window at a glance. Everything else should fade into the background, so to speak, so that the user can concentrate on the window that's active.

The mother of all alerts

The new StandardAlert routine is possibly one of the most useful routines in Appearance. It allows you to specify the text of the alert and optionally some explanatory text. You can display up to three buttons with your choice of text, as well as a help button (see **Figure 5**). The alert auto-sizes itself based on the amount of text passed into it and also auto-sizes and places the buttons. StandardAlert makes it simple to generate alerts with the standard MacOS alert look without using resources.


Ion Storm Warning

Transporting during an ion storm can cause unpredictable results. It might even throw some unlucky folks into an alternate universe. Are you sure you want to do this?

Figure 5. *Alert generated by StandardAlert.*

Moveable Modal Dialogs and Alerts

Appearance provides a standard movable modal behavior. Instead of having to write your own code for handling movable modal dialogs, you can use the new movable-modal flag in the 'dlgx' resource (or call NewFeaturesDialog). The movable-modal flag tells ModalDialog to handle all the standard user interactions, such as dragging a dialog by its title bar or

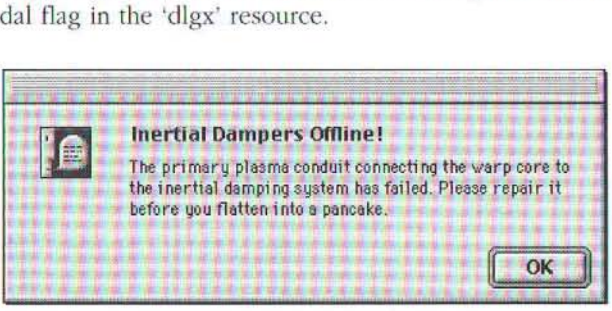
Listing 1. *Providing live feedback for a scroll bar*

```
ControlHandle CreateMyLiveScrollBar(WindowPtr window,
Rect* bounds,
  SInt16 value, SInt16 min, SInt16 max, SInt32
refCon)
{
  ControlHandle  control;

  control = NewControl(window, bounds, "\p", true,
value, min, max,
           kStdScrollBarLiveProc, refCon);
  if (control)
    SetControlAction(control,
      NewControlActionProc(MyScrollBarAction));
  return control;
}

pascal void MyScrollBarAction(ControlHandle control,
SInt16 part)
{
  SInt32  oldA5 = SetCurrentA5();

  if (part == kControlIndicatorPart)
    /* At this point, the value will be updated to properly reflect */
    /* where the user has scrolled to in our imaginary document. */
    ScrollToLocation(GetControlValue(control));
  SetA5(oldA5);
}

void TrackMyScrollBar(ControlHandle control, Point
where)
{
  /* Assuming the mouse was clicked in the indicator of a control, */
  /* you'd call TrackControl like so: */
  TrackControl(control, where, (ControlActionUPP)-1L);
  /* You could equally as well have passed the action proc in here */
  /* instead of calling SetControlAction in CreateMyLiveScrollBar. */
}
```

switching out of the application by clicking in another one. It's up to you to use the right window type (kWindowMovableModalDialogProc).

To allow your application to handle events while the dialog is up, you simply pass in a ModalFilterUPP as you do with ModalDialog. One major difference is that all events are passed to your ModalFilterUPP for handling; this allows you to handle suspend and resume events when your application is either put into the background or brought to the front, as well as any other events you might want to handle. You could use this ability to allow your application to handle Apple events from other applications even though your application is in a modal state.

It's just as easy to make your alerts take advantage of this movable modal behavior (see **Figure 6**). All it takes is a quick flip of a bit (kAlertFlagsAlertIsMovable) in the 'alrx' resource. This gives you the same behavior as setting the movable-modal flag in the 'dlgx' resource.


Inertial Dampers Offline!

The primary plasma conduit connecting the warp core to the inertial damping system has failed. Please repair it before you flatten into a pancake.

Figure 6. *Standard movable alert.*

Moving and sizing dialog items

The MoveDialogItem and SizeDialogItem routines were added to help you keep controls and dialog item rectangles in sync. If you call the old MoveControl and SizeControl routines on a dialog item, only the control is affected, making it easy to forget to make corresponding changes to the dialog item rectangle. The new routines affect both the control and its dialog item rectangle.

WINDOW MANAGER ADDITIONS

Not only does the standard WDEF add a collapse box to the title bar when Appearance is running, but the collapse/expand mechanism is improved. Windows that support the new API actually remember that they've been collapsed (except after a restart), so when you hide and show an application the collapsed state of those windows remains intact.

The tracking of the collapse box is handled by the system for you. There's currently no mechanism available to allow you to intercept tracking, but we know it's useful and are working on a way to make it available in the future. Calls to FindWindow will return the new widget's part code, so your application should be able to deal properly with part codes it doesn't understand.

Appearance offers three routines for collapsing and expanding windows: CollapseWindow, CollapseAllWindows, and IsWindowCollapsed. These routines work only on windows that actually support the new collapsing mechanism. How does Appearance know they support collapsing? As with controls, there's a new message to which windows can respond by returning a bitfield of features.

When a WDEF lets the Window Manager know that the window can be collapsed (through the feature bitfield), it can then be collapsed in the proper manner. The feature bit tells you that the window has a collapse box, which serves as an obvious widget that can be clicked. Appearance supports collapsing by double-clicking in the title bar of a window as well, but that's not as obvious to the user, so it's important to display the collapse box.

There are no new messages to make a window calculate its regions in its collapsed state. When called with a wCalcRgns message, the WDEF should check to see whether it has been collapsed with a call to IsWindowCollapsed and calculate its regions based on whatever it considers its collapsed look to be. Normally this is the title bar alone.

SOMETHING NEW ON THE MENU

Developers who use a custom MDEF may find that their menus look out of place when running with Appearance. To avoid this problem, Appearance extends the system MDEF to provide many of the features that developers have been using other MDEFs for. Now you can set the font of any menu item, making WYSIWYG font menus simple to create. You can also set extended modifiers, command IDs, text encodings, icons, and hierarchical menus, as described below. In addition to these new features, we reserved two long integers for your use, so that you can attach any other values you want to menu items. These are called, remarkably, refCon and refCon2.

You can specify the features you want for your menus either by calling the new routines mentioned below or by specifying the information in a new resource type ('xmnu'). When GetMenu is called, the Menu Manager looks for a resource of type 'xmnu' with the same resource ID as the 'MENU' resource. If the resource is found, it sets the extended information for each menu item of the menu for you. After the menu is created, you can adjust the values with the new routines.

Extended Modifiers

You can finally attach extended keyboard modifiers to menu items. So go ahead and add the Command-Shift-Option-K to your application that you've been dreaming about. You can specify the extended modifiers in the 'xmnu' resource or by calling SetMenuItemModifiers. Use the GetMenuItemModifiers routine to get the current modifiers for an item.

You might wonder how those keyboard events will get processed correctly. After all, MenuKey doesn't have a modifiers parameter. The answer is MenuEvent, a new routine that takes a pointer to an event record. It returns a long integer as MenuKey and MenuSelect do, with the low word containing the item number of the chosen command and the high word containing the menu ID of the menu containing it. If nothing was chosen, 0 is returned.

Command IDs

To help out frameworks and other technologies like OpenDoc, you can assign a command ID to any menu item. This lets you forget about the position of menu items. Instead of having to track down which menu item corresponds to a given menu ID and item number, you can use the item's command ID to identify it. Listing 2 gives an example of using command IDs. Assuming you've set a different command ID for all your menu items, you can just call GetMenuItemCommandID to get the command ID and then perform the corresponding operation. In Listing 2 this is done via a switch statement.

Text encodings

You can set the text encoding for a menu item. Think of text encodings as script codes. Previously you had to set the keyboard equivalent of a menu item to $1C and the icon ID of the item to the script code. The $1C would tip off the MDEF that there was really a script code in the icon field, not an icon ID. Now you can set the text encoding in the 'xmnu' resource or call the SetMenuItemTextEncoding routine. This enables you to have a text encoding and an icon simultaneously.

Icons

To set an icon for a menu item, you give it a handle to an 'ICON', 'SICN', or 'cicn' resource or an icon suite. The SetMenuItemIconHandle routine takes a parameter to determine the type of icon handle you're passing in, and a parameter for the icon handle itself. If you set an icon with this routine, it overrides any icon ID you may have set with SetMenuItemIcon. Using the new routine also allows you to plot 'SICN' resources and compressed (16-by-16) 'ICON' resources and still have a keyboard equivalent. Prior to Appearance, you needed to set the equivalent to $1E and $1D, for 'SICN' and 'ICON', respectively. Of course, you can also specify this information in the 'xmnu' resource.

Hierarchical menus

Appearance offers an improved method for attaching submenus to menu items. Previously, the only way to do this was to set the keyboard equivalent of the menu item to $1B and then set the mark character for the menu to the ID of the submenu you wanted to attach. Now you can use the 'xmnu' resource or the new SetMenuItemHierarchicalID routine to set the menu ID of a submenu. Both allow you to use a full 16-bit integer for your submenu ID. Freeing up the mark character makes it possible to have a checkmark next to a hierarchical menu.

DRESSING UP WITH THE APPEARANCE MANAGER

The Appearance Manager is your one-stop shopping center for getting any colors and patterns needed to draw consistently with the current theme. In the first version of Appearance, the current theme is always Apple Grayscale, but it's important to start thinking about themes. With the APIs provided with the Appearance extension, you can get colors for such things as the active window header text or the inactive menu text. There are also several patterns you can get for dialog background patterns and the like.

Getting Appearance Manager colors and patterns makes it easier to create custom defprocs for UI elements that blend with the theme. If you're not using the Dialog Manager for certain windows, these routines can help you set your text color properly so that it matches the current theme.

Along with colors and patterns, there are also routines to draw Appearance primitives. Appearance primitives (as opposed to dialog primitives) are such things as visual separators, group box lines, placards, and focus rings. The controls provided with Appearance call these routines to help their drawing. You might also call them to draw elements that match the current theme when you don't want to use a control.

In future versions of Appearance, there will be routines to do high-level things such as draw button backgrounds. Using such a routine, you could create a button with a specialized content type and be guaranteed that your button background would always draw correctly for the current theme.

ADOPTING APPEARANCE

There are varying degrees to which you can prepare for Appearance. This section covers the basic preparations and then gives some specifics on using Appearance in your application. The sample code accompanying this article is a concrete example of an application that uses Appearance, so you can peruse the code after reading this section to get a feel for how Appearance features are used in an honest-to-goodness application.

Never assume

You should never assume things about the environment

Listing 2. *Using command IDs*

```
menuID = HiWord(menuResult);
itemNo = LoWord(menuResult);
GetMenuItemCommandID(GetMHandle(menuID), itemNo,
&command);

switch (command) {
  ...
  case kCmdQuit:
    PrepareToQuit();
    break;
}
```

your application is running in. Always use routines like Gestalt and those in the Script Manager to get information about the environment. If there's no direct way to get the information, you probably shouldn't be doing whatever it is you're doing. Let's look at a couple of examples.

Window metrics

Properly determining window metrics will help you position windows correctly, no matter what the window looks like. For example, in the past you may have let your application blindly assume that the window border is 1 pixel thick or the title bar is 19 pixels high. However, the structure region of a window is controlled by the WDEF, and with Appearance it's not guaranteed to be any particular number of pixels thick. Apple Grayscale borders are thicker than the old System 7 look, and when switchable themes are implemented, borders will vary by theme. Your application should be able to deal with this intelligently by getting the structure and content rectangles for the window and using them to calculate the width of the window border. Listing 3 shows how to do this properly. The sample application has a routine to size a window and set a window's bounds; the code also shows what makes the GetWindowRects routine tick, in case you're interested.

Window variants

"Be careful what you ask for, you just might get it." Many applications ask for a documentProc window type and then never call DrawGrowIcon because the window isn't supposed to have a size box. Better to ask for the variant you really want — in this case, noGrowDocProc. As shown earlier, with Apple Grayscale the size box is part of the structure region. Because of this, the size box would be drawn automatically for a documentProc window type. We had to do some work to get around this for times when the user is running in System-wide Grayscale Appearance mode. Using the correct window variant is a step in the right direction. The Apple Grayscale WDEF has a set of variants that make it clear whether a window has a size box or not.

Appearance Savviness

Now that you've got these assumptions out of the way, becoming Appearance-savvy is really not that difficult. Just do the following:

- Call RegisterAppearanceClient early in your application code, before you draw the menu bar or create any UI elements.
- Use the new system-supplied windows, controls, and menus.
- Use the new 'dlgx' and 'alrx' resources to supplement your 'DLOG' and 'ALRT' resources.
- In dialogs, change any user items that are now available as controls (for example, a group box user item) into controls.
- Enable embedding and Appearance-savvy backgrounds in your dialogs.
- Make your alerts movable, and use the new StandardAlert routine whenever possible.
- Use the Appearance Manager to get any colors and patterns you need to draw consistently with the current theme.

The RegisterAppearanceClient routine tells Appearance that you want to map all calls to the classic defprocs (WDEF 0, for example) to the new defprocs automatically. The routines that translate from old to new form the compatibility layer, which is part of the mechanism that produces the Apple Grayscale look when "System-wide Grayscale Appearance" is checked in the Appearance Settings control panel. So calling RegisterAppearanceClient is an easy first step in adopting Appearance. To adopt Appearance completely and take advantage of some Appearance features, you have to call the new defprocs directly, as described in the following sections.

You can phase the above steps into your application at your own pace, adopting Appearance as your schedule permits. It's not absolutely necessary to do everything at once. For example, you might have some dialogs where you can add the new resource 'dlgx' type, flip the right feature bits, and be fully Appearance-savvy without doing any more work. Those would obviously be the ones to convert first. Then you can go on to other dialogs where you need to replace old user items with the new controls. When I converted the Date & Time control panel to use Appearance, I eliminated all but one of the user items (the menu bar preview) because Appearance provided controls to replace them (group boxes, icons, list boxes, and so on). So take your time, but remember that the sooner you adopt Appearance, the sooner your application will be ready for switchable themes when they're released.

When converting an application, be sure to run Appearance with System-wide Grayscale Appearance mode turned off. Turning off this mode puts your system back into the old System 7 look for applications that haven't adopted Appearance, which makes it easy for you to tell where you've implemented the new look and where you still have work to do. If you're running in System-wide Grayscale Appearance mode, you won't be able to distinguish the changes you've made from those performed automatically by the system.

COMPATIBILITY LAYER VS. DIRECT ADOPTION

As you begin to adopt Appearance directly and rely less on the compatibility layer, you'll want to change the defproc IDs for windows, controls, and menus to the new IDs listed in Table 1. Even when calling the new defprocs directly, your application should always call RegisterAppearanceClient so that system elements such as the Apple menu will draw correctly. Read on about some differences you'll encounter when calling the new defprocs directly instead of going through the compatibility layer.

Window variant codes

The variant codes for the new WDEFs are different from the

Listing 3. *Using the structure region to position windows*

```
void MoveWindowStruct(WindowPtr window, SInt16 horiz,
SInt16 vert,
  Boolean update)
{
  Rect   structRect, contRect;
  Point    structTL, contTL;
  SInt32 diff;

  /* Get the structure and content rectangles in global coordinates. */
  GetWindowRects(window, &structRect, &contRect);
  /* Calculate the difference between the top left of the content */
  /* region and the top left of the structure region. */
  structTL = topLeft(structRect);
  contTL = topLeft(contRect);
  diff = DeltaPoint(contTL, structTL);

  /* Add the difference, since MoveWindow moves based on the */
  /* top-left corner of the content region. */
  horiz += (*(Point*)&diff).h;
  vert += (*(Point*)&diff).v;
  MoveWindow(window, horiz, vert, update);
}
```

codes for the old WDEFs, so you need to change any place in your code where you rely on a window variant. Variant codes are window-specific and don't provide a generic way to determine a window's features. To remedy that, we've provided a better way to find out about window features: GetWindowFeatures. This routine is the window version of GetControlFeatures; it returns a bitfield to help determine which features a window supports. Here are the bits that are currently defined:

```
enum {
  kWindowCanGrow          = (1 << 0),
  kWindowCanZoom          = (1 << 1),
  kWindowCanCollapse      = (1 << 2),
  kWindowIsModal          = (1 << 3),
  kWindowCanGetWindowRgn= (1 << 4),
  kWindowIsAlert          = (1 << 5),
  kWindowHasTitleBar      = (1 << 6),
};
```

If a window doesn't respond to GetWindowFeatures, it's probably an old-style window and you should just use its variant code. The AppearanceHelpers library that comes with the SDK includes a set of routines to make it easier to determine a window's features using the variant code (for example, IsWindowModal).

Zooming variants

What's not apparent from Table 1 is that there are actually two window definitions now, one for windows and one for dialogs. The utility window has also been split into two, one for the normal variety, the second for the side title-bar version. We did this to clean things up a bit and add new features. In this release, we've improved the zooming variants.

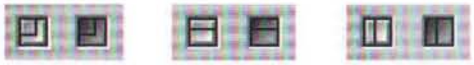


Figure 7. *Zoom box variants.*

Notice that the new defproc ID constants say "FullZoom" and not just "Zoom." It's now possible to specify horizontal, vertical, and full zoom. As a result, the zoom box is drawn differently for each variant, as shown in **Figure 7**. The part codes for the zoom box and all that you've come to know and love about zooming are still the same. The new boxes are merely a way to visually state that the zoom will act in a particular way. For example, the Apple Guide floating window used for coaching a user collapses down into a compressed version of itself (just the buttons and topic are visible) when you click the zoom box. The horizontal zoom box would be perfect to indicate how the window will zoom when clicked.

Size boxes

Size boxes in standard utility windows are now 11-by-11 pixels. The standard scroll bars can be made to fit into this space without looking scrunched, as they've been changed to support this width as well as the standard 16-pixel width. When you use old defproc IDs and go through the compatibility layer instead of calling the new utility window defprocs directly, the size box is 16-by-16 pixels.

Although not recommended, it is possible for your size box to appear elsewhere than in its new location as part of the window frame. To do this, you would use a window variant that doesn't draw a size box and then put your size box where you want it, but you'd have to handle hit testing yourself. A future version of Appearance will have a theme-savvy size box control that can be put anywhere inside a window; for now, you'll have to draw your own.

Weirdness banished

The new dialog WDEF metrics you get when adapting windows directly are slightly different from the old WDEF

Table 1. Old to new defproc ID mapping for windows, controls, and menus

Old defproc ID	New defproc ID
documentProc	kWindowGrowDocumentProc
noGrowDocProc	kWindowDocumentProc
zoomDocProc	kWindowFullZoomGrowDocumentProc
zoomNoGrow	kWindowFullZoomDocumentProc
dBoxProc	kWindowModalDialogProc
movableDBoxProc	kWindowMovableModalDialogProc
plainDBox	kWindowPlainDialogProc
altDBoxProc	kWindowShadowDialogProc
floatProc	kWindowFloatProc
floatGrowProc	kWindowFloatGrowProc
floatZoomProc	kWindowFloatFullZoomProcID
floatZoomGrowProc	kWindowFloatFullZoomGrowProcID
floatSideProc	kWindowFloatSideProcID
floatSideGrowProc	kWindowFloatSideGrowProcID
floatSideZoomProc	kWindowFloatSideFullZoomProcID
floatSideZoomGrowProc	kWindowFloatSideFullZoomGrowProcID
pushButProc	kControlPushButtonProc
checkBoxProc	kControlCheckBoxProc
radioButProc	kControlRadioButtonProc
scrollBarProc	kControlScrollBarProc
popupMenuProc	kControlPopupButtonProc
textMenuProc	kMenuStdMenuProc

metrics. Modal and movable modal variants no longer have that weird 3-pixel portion of the structure region that looked like part of the content region. This means you can finally run your content up to the edge of the window. It also means you won't have any problems with a gray background and a white border, which would happen if you erased the background gray yourself and didn't use SetWinColor to set the content color. I'm sure many of you have had to deal with this situation before.

Enabling dialog and alert features

There's a big payoff to adapting dialogs and alerts directly — you get all those great new features described previously — but it involves a little more work than adapting a normal window. To enable the new features for a dialog, you either create the dialog with NewFeaturesDialog or, for resource-based dialogs, create a 'dlgx' resource with the same ID as the dialog's 'DLOG' resource. Both NewFeaturesDialog and the 'dlgx' resource allow you to specify some flags that tell the Dialog Manager what features a dialog supports. These are the bits:

```
enum {
  kDialogFlagsUseThemeBackground = (1 << 0),
  kDialogFlagsUseControlHierarchy= (1 << 1),
  kDialogFlagsHandleMovableModal = (1 << 2),
  kDialogFlagsUseThemeControls   = (1 << 3)
};
```

The kDialogFlagsUseThemeBackground bit tells the Dialog Manager to make sure that the background of the dialog is painted in the right color or pattern for the current theme. The kDialogFlagsUseControlHierarchy bit tells the Dialog Manager to create a root control for the window and establish a control embedding hierarchy. The kDialogFlagsHandleMovableModal bit tells the system that if this dialog is frontmost when ModalDialog is called, and its window type is movable modal, it should handle the dialog as described earlier in "Movable Modal Dialogs and Alerts." Don't forget to set the kDialogFlagsUseThemeControls bit, or the Dialog Manager will create old-fashioned System 7 controls on your nice grayscale dialog. Ick.

You can make alerts Appearance-savvy just by adding the new 'alrx' resource and setting the right bits. If you want to save some code, call the new StandardAlert routine to present your alerts. If your dialogs are Rez-based, it's really easy to create new resources. The Appearance.r file included with the SDK has the Rez definitions of the new resource templates.

Using embedding

If you turn on embedding in a dialog or alert, you may need to alter your code to deal with the fact that all items in the dialog or alert are considered controls in that mode. The sample application gives examples of how to code a dialog with an embedding hierarchy. A good example is the code that demonstrates StandardAlert. The sample code puts up a dialog in which you can set the parameters to a call to StandardAlert. Because everything is a control, you can easily enable and disable editable text fields and groups of items, for example. This results in the code for that sample dialog being very small, simple, and straightforward, especially compared to what it would have been if you had to do all those things yourself!

With Appearance's new controls, you should be able to eliminate most of the user items in your dialogs. This is a fairly straightforward process of changing the user items into control items. Remember that if your dialog has an embedding hierarchy, you should change your user items to (at the very least) user pane controls. The callback to draw is practically the same.

Menus on the fly

If you create menus on the fly and want to adapt them directly, you should use the NewThemeMenu call instead of NewMenu, because NewMenu assumes MDEF 0, which forces you through the compatibility layer. You'll find NewThemeMenu in the AppearanceHelpers library.

So, what can you rely on?

One of reasons for adopting Appearance is to prepare your application for switchable themes. In an environment where the entire look of the interface can change at any moment, it might seem that you have to be ready for anything. In some respects, that's true, but there are some things that you can rely on. These aren't assumptions, they're facts!

- Default rings and focus rings can be outset a maximum of 3 pixels.
- Editable text frames, group box frames, and list box frames can be a maximum of 2 pixels thick.
- Progress indicator borders can be outset a maximum of 2 pixels.
- Metrics of controls are the same across all themes, though borders, which are drawn outside a control's rectangles, can change for default and focus rings.
- Window structure metrics actually do change.
- The menu bar height can vary, but will never be more than 24 pixels.

This information is provided to help you lay out your UI elements with enough room to look good in all themes. When theme-switching is available, an Apple event will inform your application if a theme switch occurs. Then your application can adjust window positions to accommodate changes to window frames and menu bar heights.

NOW IT'S YOUR TURN

I've tried to show you some of the highlights of Appearance, but there's only so much you can convey in words. To get a better feel for what it's all about, check out the sample application. It has a lot of useful, real-world examples of using Appearance, especially dealing with dialogs and embedding.

I think you're going to love using the new routines and features that are now available. You'll find you can get much more out of the Toolbox, which translates into less code that you have to write and less time required to implement your interface. But wait, there's more! Along with those benefits, you'll be ready for switchable themes as well. Now go check out that sample code and get cracking!

develop

by Cal Simone

According to Script: An External Editing Apple Event Protocol

Cal Simone

In recent years, Macintosh developers have begun to move away from large, monolithic programs in favor of a more modular approach to software, producing leaner, more efficient applications that don't try to do everything. An effective way to accomplish this is through external editing, in which one application uses Apple events to call upon another application for editing services. External editing allows your application to take advantage of editors implemented as Apple event–based server applications, which I discussed in develop Issue 29 that can be found at <http://17.126.23.20/dev/toc.shtml>.

In this column, I'll be exploring the advantages of implementing external editing and the basics of making it work, and I'll also present a good starting point for a general external editing implementation. At this time, there is no generally applicable external editing suite that's part of the Apple Event Registry. However, there is a suite in fairly wide use that's designed specifically for external editing of graphic objects in word processing documents (discussed later), and the more general external editing suite presented here is largely derived from it.

Types of Apple Event Usage

First let's look at the types of Apple event usage to see how external editing fits in. In general, there are three ways Apple events can be used:

- Direct program-to-program communication between two applications — One or both applications have intimate knowledge of the other's Apple event protocol. Either or both of the applications can be the client or the server. This usage has largely faded from practice.
- Service modules — A client application calls upon the services of another application as though that application were embedded in the client. An example is the relationship between a word processor and a spell checker. External editing falls in this category.
- Scripting — Lots of scriptable applications are hanging out, saying "I'm available" and "This is what I do." The scripting language is the client, and it doesn't have to determine which parts of the Apple event protocol of the scriptable applications are necessary to control them; that's the scripter's responsibility.

Advantages of External Editing

There are several advantages to implementing external editing:

- Developers can effectively extend the capabilities of their applications by leveraging off of applications from other vendors, so it's not necessary to implement an editor for each data type supported by your application.
- Since each developer has a smaller code base to manage, applications will have fewer bugs.

Cal Simone (mainevent@his.com) has been fighting the good fight to help developers implement scriptability, cleanly and consistently, for nearly five years. He's currently excited about external editing, and if you are, too, he invites you to e-mail him to continue this discussion. He also encourages you to submit the dictionaries of your applications, parts, or scripting additions to review@script.org.

- Users can pick and choose the combination of tools they like from different vendors, benefiting from the advantages that a stand-alone editor designed for specialized usage can offer, while retaining access to all the features of the client application.
- Users can use one environment for editing the same kind of data, regardless of which applications they work in, enjoying a consistent experience.

If the above sound familiar, that's because these same advantages are central to OpenDoc's philosophy. In the non-OpenDoc world, you can achieve much the same effect by implementing external editing.

Note that in this discussion, client means the application requesting the edit (not the actual user), and editor means the server application. Any application implementing the Apple event object model and the Core suite can easily take advantage of external editing as a client, since most of the work is done by the server.

An Eventful History

The work done in the Word Services Apple event suite paved the way for a consistent services protocol. However, the details of that suite are somewhat specific to spell-checking and similar service modules, where the client and server interact more intimately. A modified version of the EGO (Edit Graphic Object) protocol is better suited to external editing's needs. The main difference between EGO and the external editing protocol that I describe here is that external editing relies less on custom Apple events and parameters than on existing events defined in the Apple Event Registry, in particular the Core suite, and can therefore be quicker to implement.

In EGO, the user double-clicks on an object in a client application, a server application appears with the object in an editing window, the user edits the object and dismisses the window, and the edited version appears in place in the client's window. EGO was originally developed so that mathematical equations in word processing documents could be edited in a full-featured equation editor. However, it was designed for more general use as an external editing mechanism for graphics embedded in text documents. The first shipping implementation in 1991 was used to link Expressionist to Microsoft Word. EGO and extended versions of EGO have been implemented in word processors, such as WordPerfect, Nisus, and FullWrite, as well as other types of applications, such as Theorist, SigmaPlot, and Chem3D. (See <http://www.well.com/~bonadio/aba/ego.html> for more details on EGO.)

Although not a part of the Apple Event Registry, the external editing protocol that I describe here, and a version that included extensions applicable to script editing, are based on EGO concepts and were developed in 1993. These ideas were shaped in discussions between Allan Bonadio, founder of Prescience, Michael Rubenstein of CambridgeSoft, Lee Buck of Software Designs Unlimited, and myself, and were reviewed by others. In 1995, a portion of the scripting version was implemented in FaceSpan and Scripter to allow editing and debugging of scripts in a stand-alone script editor. Currently shipping versions of third-party stand-alone script editors include some form of the protocol, with extensions or variations.

Please note that to make the external editing protocol useful for as wide a range of data, clients, and editors as possible, I haven't included the parts of EGO or its variations that are specific to editing particular data types. I'm going to concentrate here on general-case scenarios and the Apple event protocols — the operations that can form the basis for external editing. This is not a complete treatment of external editing by any means; in particular, I'll mostly steer away from user interface issues and discussions of error handling, both of which are important issues that need to be addressed in any complete external editing suite. The goal of this column is to serve as a catalyst to speed the creation and adoption of such a suite.

To be Modal or Non-Modal?

The external editing protocol was designed to allow both modal and non-modal editing. In modal editing, the client application requests the services of the editor and waits for the user to finish editing. The editor returns the edited item to the client, and the client resumes operation. This modal editing situation prevents the user from working in the client application during the editing session. I highly discourage anyone from going this (user-unfriendly) route. The freedom, ease, and elegance of non-modal editing far outweigh any superficial advantages of modal editing. Modal editing is not recommended and won't be discussed further here.

In the less rigid world of non-modal editing, the client application requests the services of the editor and allows the user to edit asynchronously. During the editing session, the client uses whatever version of the object it has, either the original or an updated version. When the editing is done, the editor replaces the object. As optional parts of the protocol, the client can request the return of the currently edited version at any time or cause the editing session to be abandoned.

What You Need to do

If you're already supporting the object model and the Core suite, all you need to do is write the code to send (for clients) or handle (for editors) one new Apple event, and optionally handle (for clients) or send (for editors) another new Apple event. If you don't support the object model, this is a good excuse to get started; you just have to write the additional code to support three Core events that are used in the external editing protocol. Now let's take a look at how all this can be accomplished.

Initiating the edit

The user requests an edit, through some action in the user interface of the client application. The mechanism by which a user initiates an editing session is not specified as part of the protocol. You can decide what's appropriate for your application — possibilities include double-clicking (perhaps with a modifier key held down) on an object in one of the client's windows, clicking a tool in a palette, or choosing a menu item or button to act on the current selection.

The client application requests an editing session by sending an Edit Apple event to the editor application. The Edit event, summarized in Table 1, is similar to the Edit Graphic event in the

Table 1. Edit Apple event and parameters

Event	**Suite and event IDs**	**Meaning**
Edit	'edit', 'edit'	Sent from the client to the editor to initiate an editing session.
Parameter	**Type**	**Meaning**
Direct parameter (keyDirectObject)	Object specifier (or data to be edited)	The client's reference to the item to be edited; this is the only required parameter.
keyAEName ('pnam')	String	The title for the editor's editing window. (This parameter is highly recommended.)
keyAERequestedType ('rtyp')	Class type	The type of data the client wants returned when the edit is the edit is done.
keyAEPosition ('kpos') or keyAEBounds ('pbnd')	Point or bounding rectangle	The position or bounds for the editing window.
keyAENotifyWhenDone ('Noti')	Boolean or reference	The client wants to be notified when the userterminates the edit. If the value is Boolean,notify the application that sent the event; ifa reference, notify the application in the reference.
keyAESendHeartbeat ('Beat')	Boolean	The editor should check periodically for the presence of the original item.
keyAEUnique ('Uniq')	Integer	The unique ID of a specific editing session.
Reply parameter	**Type**	**Meaning**
Direct parameter	Object specifier	The editor's reference to the newly created editing session.

Miscellaneous Standards of the Apple Event Registry.

The direct parameter of the Edit event is an object specifier, which is a reference to the item to be edited. I'll call this the client's *original item reference*. You can use any form of object specifier that suits you, although it's important to create an object specifier that will always refer to that specific object regardless of the state of the application — formUniqueID fits the bill. Note that both formAbsolutePosition and formName should be avoided as object specifiers in this case, because references by index can change as items are rearranged while the edit is occurring, and names may not be unique and can be changed at any time.

It's also a good idea to include a name parameter (using the keyAEName keyword), which is a text string that the editor can use as the title for the editing window. Usually this is the name of the item that's being edited, although you can make it something else, if appropriate.

The client can include an optional return type parameter (using the keyAERequestedType keyword) to specify the type for the updated data when it's sent back to the client after editing. In addition, the client might want an editing window to be positioned at the same place on the screen as the original item appears in the client's window. To do this, the client can include an optional parameter (using the keyAEPosition or keyAEBounds keyword) indicating the position or bounds that the editor can use to determine the location for the editing window. The value of this parameter could be the client's global screen coordinates of the visible object.

Finally, it's a good idea to include a session ID parameter (using the keyAEUnique keyword, with the ID 'Uniq'). The session ID parameter value is a long integer of the client's choosing, such as TickCount, that uniquely identifies the editing session.

If the attempt to send the Edit event results in the error procNotFound, the editor isn't running. If this happens, the client should launch the editor and send the Apple event again.

The editor receives the client's request for an editing session and handles the Edit event, causing the following actions to occur:

1. The editor extracts the original item reference from the direct parameter (and the session ID parameter value, if the client included it) and keeps this value with the data for the editing session.
2. In the direct parameter of the reply Apple event, the editor places a new object specifier, which is a reference to the editing session. I'll call this the editing session reference. The editor then returns from handling the Edit event.
3. The editor requests the item to be edited from the client by sending the client a Get Data Apple event, passing the client's original item reference as the direct parameter.
4. The client responds to the Get Data event by resolving the original item reference in the direct parameter (which the client itself originally generated), inserting the data for the item to be edited in the direct parameter of the reply, and returning from handling the Get Data event.
5. The editor extracts the item's data from the reply to the Get Data event, opens an editing window, and places this data in the window.

6. The editor is brought to the foreground, ready to edit the data. If the client launched the editor and requested that it be brought to the foreground, this will happen automatically; otherwise the editor can bring itself to the foreground. (How to do this is discussed in my last column, and as I said there, this is just about the only situation where it's reasonable for a server application to force itself into the foreground without asking the user to switch applications.)

Finally, both applications return to their main event loops, with the editor now in front. The appropriate suspend/resume and activate events are delivered by the system and handled by both applications as expected.

Early versions of the Apple Event Manager imposed a 64K limit on the size of data in an Apple event, including the attributes and any parameter data. Your application should be prepared to deal properly with this. The limit is not an issue if AppleScript 1.1 is running, and the limit has been completely removed in MacOS 7.6.

Performing the edit

The user performs the desired edits in the editor application. In this non-modal situation, the user can switch to other programs, including the client application, while editing. The user can even initiate an editing session for a different item from the same client — the original item references and the editing session references keep things straight.

Generally speaking, while the user is editing an item in the editor, the client should not allow any action that would cause that item to be modified. If the user again requests an edit of the same item, the client should send an Edit event that is identical to the original edit request, with the same object specifier and session ID parameter. Under normal circumstances the editor should not initiate another editing session for that item. The correct behavior is for the editor to activate the editing window corresponding to the original item.

Terminating the edit (from the editor)

The user finishes editing by closing the editing window in the editor application. If there is changed data, then before the window is actually dismissed the editor should display an alert, asking the user to confirm that the changes should be sent back to the client. If the user so confirms, the editor creates and sends a Set Data Apple event to the client application, passing the client's original item reference in the direct parameter, and the updated item data in the data parameter (using the keyAEData keyword). If the client included a session ID parameter with the original edit request, this parameter is also included in the Set Data event.

The client application responds by resolving the original item reference and extracting the updated data from the data parameter. The client can replace the original item with the updated data or otherwise use the data in whatever manner it sees fit, such as updating a database or broadcasting the change to other applications. The presence of the session ID parameter tells the client that this Set Data event is associated with the editing session, as opposed to any other Set Data event that might change the data for the original item. After the client returns from handling the Set Data event, the editor may safely dismiss the editing window.

If appropriate, the client could now be brought to the foreground. At this point both applications have again returned to their respective main event loops, and the user may continue working, using the new version of the edited item. The user has had a smooth editing experience and is happy.

EXTEND YOUR OPTIONS

In addition to the basic process described above, there are several options that may be implemented by either the client or the editor. Implementing these options requires only that you send no more than three additional Core Apple events (two of which are Apple events you must handle anyway) and handle one new Apple event.

Working with interim versions of the edit. During the editing session, the user may want to try out versions of the edited item without terminating the editing session. Here's one way to accommodate this:

1. The user requests an interim version be sent to the client application (by choosing a Send Back command like that in Scripter, for example, or perhaps by choosing Save).
2. The editor creates a Set Data Apple event containing the current version of the edited item's data, the same as it does when terminating the edit, and sends the Apple event to the client application.
3. The client application extracts the updated data from the Apple event.
4. The user tries out the updated version of the item.

In general, the client application will know when an updated version of the item is received via a Set Data event by the presence of the session ID parameter, but it may not know when the editing session is completely finished. If the client does need to know when the edit is finished, in order to stop keeping track of whether the item has an outstanding editing session, the client can request that the editor notify the client when the editing session is finally done. The editing completion notification request is made when the client first initiates the external editing process (more on this below).

Calling the edit back (terminating the edit from the client)

If the client application allows it, a user may (while in the middle of editing) delete or copy the original item in the client application, close the window containing the original item, or quit the client application. In any of these situations, the client application may need to obtain the currently edited version of the item and/or terminate the editing session. Essentially, the data in any outstanding editing session is an extension of the client's application or document data at large and must be called back for any reason that requires the current value of the item's data.

Note that the act of closing the server's window is not required, but it is highly recommended. Leaving the editing window open creates a situation where later closing the window and confirming the changes could result in problems. The editor might attempt to send a Set Data Apple event to an application that's no longer running, or with an object reference that's no longer valid. Even worse, the object reference might be valid, but now refers to a different object (depending on the nature of the object specifier).

To call an edit back and terminate the edit:

1. The client terminates the editor's editing session by sending the editor a Close Apple event, with the editing session's reference in the direct parameter and an optional kAESaveOptions parameter. The value of kAESaveOptions determines whether the updated data is called back to the client; the default value is kAEAsk.
2. The editor receives the Close event, extracts and resolves the editing session reference, and extracts the keyAESaveOptions parameter, if present. If the value of keyAESaveOptions is kAENo (sent when the client isn't interested in the updated data), the editor returns noErr as the result of its event handler, closes the window, discards the data, and skips the remaining steps. If the value is kAEYes, the editor proceeds to step 4.
3. If the editor finds a keyAESaveOptions parameter with a value of kAEAsk, it displays an alert confirming that changes should be sent back to the client. If the user clicks Cancel, the editor returns userCanceledErr as the result of handling the Close event, the client abandons the current operation, and the remaining steps are skipped. If the user clicks Don't Send, the editor discards the edited data, closes the editing window, and returns noErr as the result of handling the Close event, and the remaining steps are skipped.
4. The editor creates a Set Data event containing the current version of the item's data and sends this event to the client. The client must implement idle and filter procs to handle the incoming Set Data event while the Close event is outstanding, otherwise this will result in an Apple event deadlock.
5. The client checks the result of sending the Close event. If the result is userCanceledErr, the client cannot quit or close the window containing the original item.
6. The editor discards the edited data and closes the window.

Reverting to a previous version (by the editor)

While working in the editor, the user may want to revert to a previous version of the edited item. If the user chooses Revert from the editor's File menu, the editor should display a standard "Revert to last version?" alert, then replace the altered version with the original version from the client. To revert to the client's version:

1. The editor requests the previous version of the edited item by sending a Get Data Apple event to the client, with the original item reference in the direct parameter.
2. The client replies with its current version of the item's data in the direct parameter of the reply to the Get Data event.
3. The editor extracts the earlier version from the direct parameter of the reply and displays it in the editing window, in place of the edited version. The user may now continue with the editing session.

Checking the status of an edit (by the client)

If the client wants to know whether the editing session is still in progress, it can send a Does Object Exist Apple event to the editor, with the editing session reference in the direct parameter. If the reply contains a Boolean value of false or if you get a procNotFound error, it's no longer a good idea for the client to send a Close or Get Data event for the editing session; the client can conclude that the editing session has been abandoned or the editor has crashed. The client can deal with the possibility of not receiving a completed edit by notifying the user with an alert explaining that the editing session may have been abandoned.

Table 2. Editing Is Done Apple event and parameters

Event	**Suite and event IDs**	**Meaning**
Editing Is Done	'edit', 'done'	Sent by the editor to the client when the editing session is finished, if notification was requested by the client.
Parameter	**Type**	**Meaning**
Direct parameter	Object specifier	The editor's reference to the editing session.
keyAEUnique ('Uniq')	Integer	The unique ID of the specific editing session (required if the client included this parameter in the original edit request).
Reply parameter	**Type**	**Meaning**
—	—	No reply from the client.

Checking the status of an edit (by the editor)

Sometimes it can be helpful for the editor to know whether the client is able to receive the results of a completed edit. The editor can check just before it sends back the updated data for the item by sending a Does Object Exist Apple event to the client application, with the client's original item reference in the direct parameter. If the editor doesn't successfully receive a reply (or receives a reply with a Boolean value of false) to the Does Object Exist event, the editor can assume that conditions in the client application for receiving updated item data are unfavorable and that it should not attempt to send back updated versions of the item. The editor can notify the user by putting up an alert, asking if the edited version should be saved to a file or abandoned.

To determine whether an editing session is still valid, the editor can check periodically for the existence of the client's original item on which the edit is based. One way to do this is with a heartbeat, which is requested by the client when the edit is initiated by including the optional heartbeat parameter (using a keyAESendHeartbeat keyword, with the ID 'Beat') in the Edit Apple event.

Notification of editing completion

If an original item still has outstanding edits, it may not be safe to close the window, delete the item, or quit the client application. A client application that's keeping track of outstanding edits can optionally request that the editor notify it when the editing session is finished.

The client requests notification of editing completion when the edit is initiated, by including the optional notify parameter (using a keyAENotifyWhenDone keyword, with the ID 'noti') in the Edit Apple event. Then, when the user closes the window or quits the editor, and after the final Set Data event, if any, has been sent to the client, the editor will send an Editing Is Done Apple event (necessary even if no changes were made in the editing session). The Editing Is Done event is summarized in Table 2. The client responds to this event by noting internally that the editing session is completed.

Sending the original data to be edited in the edit request

There are situations where there's no object to be edited in the client application, such as when the client wants to request editing for data that's in a file on disk. In situations where the client isn't directly concerned with the editing session, it can send an Open Apple event to the editor with a file reference or alias to the file to be edited in the direct parameter. This is proper when the data to be edited isn't embedded in a client's document or application data, and the client isn't responsible for storing the updated data; the editor takes care of this.

On the other hand, when parts of a document are stored in separate files, as with a page layout application, it's better that the client use the normal methods outlined above, creating Edit Apple events and object specifiers. While it is possible to send the actual data for the item to be edited in the direct parameter of the Edit Apple event, I generally don't recommend this approach. The solution is to support the object model; in this day and age, this is simply the best way.

Be a Pioneer

Obviously, every aspect of the client/editor relationship hasn't been explored here. As you transform your application into a client or editor, you'll run into user interface issues that need to be worked out, and you'll need to implement appropriate and robust error handling. But right now, with just a few bits of code, you can begin to implement the external editing protocol as either a client or an editor and try it out. If you don't yet support the object model, this is your chance to get with the program.

Talk to other developers whose applications you want to work with, and get the discussion going. A good place to start is the applescript-implementors mailing list (to join, see <http://www.solutions.apple.com/ListAdmin/>). With a little luck, and a little work, we'll soon have a complete specification that works for most everyone. Users will thank us for it, as they'll be able to experience the richness of using our applications in a variety of new situations.

Thanks to Andy Bachorski, Allan Bonadio, and Jon Pugh for reviewing this column. Allan generously donated the use of the EGO concepts, and Michael Rubenstein insistently convinced the author to look at EGO. Lee Buck worked with the author to simplify the events.

RELATED READING

- *Inside Macintosh: Interapplication Communication* by Apple Computer, Inc. (Addison-Wesley, 1993).
- *Apple Event Registry: Standard Suites* (Apple Computer, Inc., 1992). Available on ftp in the folder ftp://ftp.apple.com/devworld/Development_Kits/AppleScript/Documentation/.
- For more information on EGO, see the Web page at http://www.well.com/~bonadio/aba/ego.html.
- For more information on AppleScript in general and to connect with other AppleScript developers, check out the AppleScript Language Association's Web page, at http://www.script.org.

by Mike Fullerton

Squashing Memory Leaks With TidyHeap

Mike Fullerton

One of the more mundane aspects of application development is making sure any system resources that your application allocates get deallocated, particularly memory allocations. Think of the hours you've spent tracking down pesky memory leaks. A powerful debugging and QA tool, TidyHeap watches over memory allocations and helps you track down leaky blocks. Learn how to make sure that memory allocated by your application is always managed correctly.

TidyHeap is a tool for tracking and verifying dynamically allocated memory in C++ applications. It started out as a quick and dirty way to keep track of undeleted objects in the MacApp framework and evolved into a tool for anyone writing application software in C++. TidyHeap works with any application built in C++, with or without a framework. Among its powerful features are the following:

- If your application allocates an object or data that's never deleted, TidyHeap tells you about it when your application quits. (Objects and data allocated by your application are called *client objects* in this article.)
- TidyHeap can provide you with the filename and line number where any existing client object was created, including a client object that's never deleted.
- You can tell TidyHeap to watch for double deletions on all client objects or on a specific client object.
- You can verify the integrity of a single client object or all client objects at each pass through the event loop.
- You can have TidyHeap break into MacsBug when a specific client object is deleted.
- You can force every single **new** operation to fail in consecutive order, in consecutive runs of your application.
- You can design your own tests for specific client objects or for all client objects.

In short, TidyHeap is designed to be flexible, powerful, and user extensible. Currently, TidyHeap runs only on the PowerPC™ platform (if you would like to see it implemented for 680x0 machines, send e-mail to mabugs@devtools.apple.com), and it's compatible only with Metrowerks CodeWarrior (we MacAppsters plan to make it compatible with the various MPW compilers sometime soon). A package of TidyHeap materials is available at <http://www.mactech.com>.

TIDYHEAP BASICS

TidyHeap itself is a small library of C++ classes that intercept and track all allocations and deallocations performed by the global operators **new** and **delete.** It does this invisibly — you don't need to do anything but call **new** and **delete** in the way you usually do. Also, since TidyHeap is essentially a debugging tool, it's compiled and enabled only if its preprocessor constant qTidyHeap is defined to true. This enables you to turn it off easily when building nondebug versions of your application.

TidyHeap works with any data type that can be created with **new.** This can be all the objects in your application or framework, structs created to be passed to Toolbox routines, or huge **char** arrays created as buffers. It really doesn't matter. TidyHeap treats all allocations as raw blocks of allocated memory, without knowing or caring what the user data is.

Mike Fullerton (mikef@apple.com) began his high-tech training when knee-high to a programmer by designing and building vast civilizations out of Legos, masking tape, and rubber bands. Believing that computers should be as much fun, he began programming in the entertainment industry. He was lead programmer for the strategy game Allied General, and then for the Macintosh version of Panzer General. While leading the development of a cross-platform C++ framework on which the games were built, he decided that making software was a pain in the brain, and he set out on a quest to make it easier. Now at Apple in the Frameworks group, his grail is making MacApp insanely great, and he's currently up to his eyeballs in providing it with networking support.

The raw materials

TidyHeap ships with two CodeWarrior projects and every source file needed to build and use it. The projects are for building the shared library, TidyHeapSharedLib, and the static library, TidyHeapPPC_d. Both libraries are required because TidyHeapSharedLib doesn't contain TidyHeap itself, but is the mechanism I use to make sure that TidyHeap is the last thing to be deleted, after static destruction time.

It's essential that TidyHeap be the very last thing deleted when the client application quits. This guarantees that the application won't try to delete anything after TidyHeap, which could cause TidyHeap to return inaccurate information. To make sure that it terminates after static destruction time, TidyHeap registers a callback in TidyHeapSharedLib that's called when the shared library terminates. It so happens that this occurs after static destruction time, which ensures that TidyHeap is the very last thing deleted.

You may build the shared library if you need to; however, a built version of TidyHeapSharedLib is also included. Install this as you would any shared library, either in the same folder as the application or in your Extensions folder — just be careful not to install it in both places. Also, don't forget to add TidyHeapSharedLib to your application's project! At this time weak importing of the shared library is not supported, so don't use weak linking. If you do, your application may crash.

The second CodeWarrior project is for TidyHeap itself. There are about 35 source files. You can include them directly in your application's project or you can build them into a static library with the TidyHeapPPC_d.π project and include the static library in your project. The choice is yours. Personally, I prefer building TidyHeap as a library.

Finally, TidyHeap ships with two small tools — New Dropper and TidyHeap Director. New Dropper allows you to easily insert into or remove from your source files an important macro, and TidyHeap Director lets you change TidyHeap's behavior without recompiling your code. These tools aren't yet fully supported by Apple (so use them at your own risk), but they're very useful — as you'll see later.

Hooking in your framework or project

Hooking into TidyHeap is simple. You define a C function named .TidyHeap declares and calls the function but doesn't define it.

(If this function isn't defined, you'll get a link error.) defines the global operators new and delete that TidyHeap calls, and allocates AC_CTidyHeap or a client-defined subclass of TidyHeap. (See "MacApp Naming Changes" for the reasons behind TidyHeap's naming conventions.)

TidyHeap has static pointers to the global operators new and delete. By default, the static pointers point to NewPtr and DisposePtr. These are the functions that TidyHeap calls to actually create and delete memory blocks. The function must set the proper memory allocation and deallocation functions for your framework before it creates AC_CTidyHeap or its subclass. For example, the following code defines for the MacApp framework in which MAOperatorNew and MAOperatorDelete are MacApp memory allocation and deallocation hooks. (Although this and later examples use MacApp, don't forget that TidyHeap works with any framework.)

```
AC_CTidyHeap* CreateTidyHeap()
{
  AC_CTidyHeap::GlobalNew = MAOperatorNew;
  AC_CTidyHeap::GlobalDelete = MAOperatorDelete;
  return TH_new TMacAppTidyHeap();
}
```

The AC_CTidyHeap class includes virtual functions that you can override if you want to change TidyHeap's behavior. For example, the method HandleNilAllocation is called by TidyHeap when it forces **new** to fail. If your framework throws an exception when **new** fails, you can override HandleNilAllocation to throw the same exception. This enables TidyHeap to properly emulate what happens when new fails in your environment. (More on forcing **new** to fail later.)

To use the default TidyHeap and the default memory allocation hooks, include the file CreateDefaultTidyHeap_AC.cp, which defines a function that uses the defaults:

```
AC_CTidyHeap* CreateTidyHeap()
{
  return new AC_CTidyHeap;
}
```

Now you're ready to use TidyHeap and begin reducing those hours spent tracking memory leaks.

First contact

In this section you'll meet the headers and trailers, the AC_CTidyHeapBlock object, and the TH_new macro, all of which TidyHeap uses to track and store data it needs to watch over your memory allocations and perform various actions.

TidyHeap itself is a single object that's created when the global operator new or the AC_CTidyHeap static member function Instance is called for the first time. Instance returns a pointer to the AC_CTidyHeap object. If this object hasn't been allocated yet, Instance creates it and returns the pointer. In this way you're guaranteed to have a valid pointer to the AC_CTidyHeap object. You can use Instance to call AC_CTidyHeap's interface. For example, the following line of code returns the total amount of memory currently allocated by your application using the new operator.

```
long sizeTally = AC_CTidyHeap::Instance()->GetSizeTally();
```

Header and trailers

When your application calls new to allocate a client object, TidyHeap intercepts the call and increases the amount of memory allocated by a small amount in order to put a header and trailer around the client object's memory block. TidyHeap stores data in the header and trailer that allows it to track your client object. The header contains a pointer to an AC_CTidyHeapBlock object, which has been allocated by TidyHeap and stored in a linked list. This is how TidyHeap tracks all the client objects. Every client object allocated with new has its own AC_CTidyHeapBlock object (described in the next section).

The header and trailer also contain three markers. Markers are long integers that TidyHeap sets to magic values you can easily recognize when looking at a raw memory dump of your object — 0xF1F1F1F1, 0xF2F2F2F2, and 0xF3F3F3F3. These markers act as fences around your client object (TidyHeap operates on the assumption that if the markers get trashed your client object is probably trashed also). TidyHeap can quickly verify the integrity of your block of data by checking the markers. There are two markers in the header, one just before and one just after the pointer to the client object's AC_CTidyHeapBlock object, and one marker in the trailer, as shown in **Figure 1**.

AC_CTidyHeapBlock and block IDs

The AC_CTidyHeapBlock class is important. An AC_CTidyHeapBlock object holds information about the client object your application allocated, and member functions defined for the class are used to perform various operations on the client object.

Your client object's block ID is one of the more useful pieces of information you can obtain from AC_CTidyHeapBlock. The *block ID* is a unique number assigned by TidyHeap to each client object that's created with new. The block ID corresponds to the number of times new has been called, so if the block ID is 6, the client object was created on the sixth call to new. Here's an example of how to get the block ID from a client object:

```
TClientObject* myObject = new TClientObject;
AC_CTidyHeapBlock* objBlock = AC_CTidyHeap::Instance()-
>GetBlock(myObject);
long blockId = objBlock->GetID();
```

Among the member functions defined for the AC_CTidyHeapBlock class is the Verify method. By first getting the pointer to its AC_CTidyHeapBlock object and then calling AC_CTidyHeapBlock::Verify, you can verify the integrity of your client object. The Verify method cross-references the pointer to the client object's AC_CTidyHeapBlock object that the client object stores in its header with the pointer to its client object that the AC_CTidyHeapBlock object stores. It also checks the validity of the block's markers. For example, if you write past the end of your object, the trailer marker will be invalid and TidyHeap will notify you by returning an error from Verify. Here's an example of using the Verify method:

```
TClientObject* myObject = new TClientObject;
AC_CTidyHeapBlock* objBlock = AC_CTidyHeap::Instance()-
>GetBlock(myObject);
AC_CTidyHeapError err = objBlock->Verify();
```

Storing filename and line number with TH_new

TidyHeap provides TH_new, an optional macro replacement for the global operator **new.** TH_new stores the filename and line

MACAPP NAMING CHANGES

The MacApp team is working to make MacApp a less monolithic framework. Whenever practical and possible, we're designing and implementing MacApp's new features so that they can be used independently of MacApp. TidyHeap is a prime example of this trend. Other features in development are networking support, cooperative and multiprocessing threading support, OpenDoc container support (ALOE, the Apple Library for Object Embedding), and such useful things as enhanced Standard Template Library (STL) support and a handful of C++ design patterns (useful abstractions for common problems). We're calling this collection of features "Apple Classes," but that may not be the final name.

We want everyone to be able to use Apple Classes without running into namespace and file inclusion collisions, so after much heated debate, we decided to adopt and adapt the OpenDoc Development Framework (ODF) naming convention until we get universal support for C++ namespaces in all our compilers. That's why all the TidyHeap classes are named AC_CClassName. The "AC" stands for "Apple Classes" and represents the scope of the class (ODF begins names with "FW" for "Framework"). The underscore conceptually represents the C++ scope operator (::). (For example, AC_CTidyHeap will become AppleClasses::CTidyHeap when C++ namespaces are supported.) The second "C" in "AC_C" represents the usage of the class — "C" for classes, "M" for mixin classes, and "S" for structs.

Filenames in Apple Classes significantly diverge from the ODF naming convention. ODF is cross-platform (Windows and Macintosh), so all its files conform to the Windows 8-character maximum. Because we don't need to do this, we decided to make the filename the same as the class name with a slight twist, which is that we append "_AC" to the class name like this: CTidyHeap_AC.h. We put the difficult-to-type underscore at the end to make life easier in dealing with the files in the Finder and for situations where you would want to use type-ahead, such as in a CodeWarrior project window.

If the TidyHeap classes accompanying this article begin with letters other than "AC," it's because a different name for the collection of features referred to here as Apple Classes was adopted after this article was published.

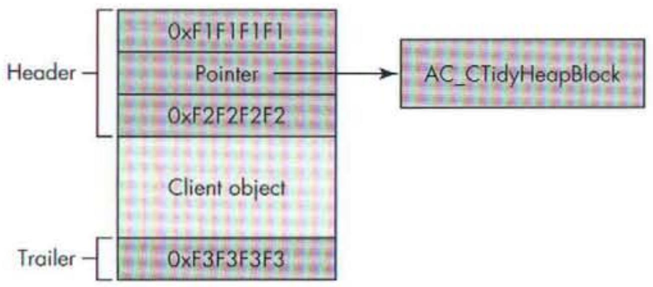


Figure 1. *Client object memory block.*

number of where the client object was created in the client object's AC_CTidyHeapBlock object. The filename and line number enable you to find the code that created undeleted client objects. TH_new uses the C++ preprocessor's built-in __FILE__ and __LINE__ macros, which is why TH_new needed to be a macro. Of course, you don't want to ship your application with TidyHeap enabled, so TH_new resolves to new when qTidyHeap is not defined or is defined to false, making it possible to leave the TH_new macro in your shipping code.

Listing 1 shows how to get a filename and line number for an object created with TH_new. When the program is run, you get the following output:

```
MyObject was created in file: Main.cp at line #1
```

The New Dropper tool that ships with TidyHeap makes it easy to switch between **new** and TH_new. Drag your project's folder onto the New Dropper tool and you're presented with two options: change TH_new to **new** or change **new** to TH_new. That's it. When you make a choice, New Dropper scans the source files you dropped on it and makes the chosen change. New Dropper finds all the source files in all the folders, no matter how deeply you've nested them. It only opens files that it decides are text files with source code in them. In other words, it won't open your project files, or even a text file that doesn't have a ".h" or ".cp" appended to its name. It doesn't change new in comments, in object-specific overrides of the new operator in a client object, or if the file has "TidyHeap" anywhere in its name. The software is provided as is, so please make a backup of your source before using it!

Warnings, logging, and the log file

TidyHeap has two types of messages — a logged message and a warning — and provides a default mechanism for directing logged messages to a log file. The log file is written to the same folder as your application and is named myapplication.log. It's an MPW text file and is compatible with MPW scripting. The AC_CTidyHeap method you can use to log a string is

```
AC_CTidyHeap::Logf(const char*, ...);
```

All the output from Logf is appended to the log file, and it's designed to be compatible with the ANSI-standard **printf.** Here's some code showing how you could use Logf:

```
TClientObject* myObject = TH_new TClientObject;
AC_CTidyHeapBlock* objBlock = AC_CTidyHeap::Instance()-
>GetBlock(myObject);
AC_CTidyHeap::Instance()->Logf("The object I just created has
an ID of %ld",
  objBlock->GetID());
```

The warning mechanism is simply a wrapper on top of DebugStr that can be turned on and off. You'll get a TidyHeap warning in only a few situations — in general, to notify you that something has gone wrong. For example, if your application terminates and there are five client objects that haven't been deleted, you'll get the warning "TidyHeap Warning: You have 5 undeleted blocks."

This undeleted-block warning has a g appended to it so that your application will continue immediately after dropping into MacsBug. I think you'll like it better than typing g and Return a thousand times.

The AC_CTidyHeap method that you use to send a warning is

```
AC_CTidyHeap::Warnf(const char*, ...);
```

Like Logf, it behaves the same way as the ANSI **printf,** except that the output goes to MacsBug, of course.

When you get a warning, TidyHeap usually logs more information about the problem to the log file, which is described next.

Using the log file

Let's see how to use the log file to track down memory leaks. For example, after running the program in Listing 2, you'll get this warning from TidyHeap: "TidyHeap Warning: You have 2

Listing 1. *Getting the filename and line number*

```
void main()
{
  TClientObject* myObject = TH_new TClientObject;
  AC_CTidyHeapBlock* objBlock =
    AC_CTidyHeap::Instance()->GetBlock(myObject);
  const char* fileNamePtr = objBlock->GetFileName();
  long lineNum = objBlock->GetLineNum();
  printf("MyObject was created in file: %s at line
#%ld\n",
    fileNamePtr, lineNum);
}
```

undeleted blocks." When there are undeleted client objects after your application quits, TidyHeap logs information about all the objects that haven't been deleted to the log file.

Here's what the log file looks like after you run the program in Listing 2:

```
### TidyHeap Termination Messages:
### 2 Outstanding blocks, 3 total blocks, Ave Size 4.500000
### New called 3 times, Delete called 1 times
### BlockInfo: ID:1, Address:0x1EC8F8C, Size:4
        File 'Main.cp'; Line 6
### BlockInfo: ID:2, Address:0x1EC902C, Size:4
        File 'unknown'; Line 0
### Sum of outstanding block sizes = 8 bytes
```

The log file contains some useful information, as you can see. There are two outstanding blocks — the two TClientObject objects that the program in Listing 2 neglected to delete. The **new** operator was called three times, and **delete** was called once, which jibes with what we know from the example code.

Because **object1** was created by the TH_new macro, you can see exactly where it was created and go back and make sure it gets deleted — a piece of cake, even on huge projects.

Unfortunately, we didn't use TH_new for **object2,** so TidyHeap logged its location as "unknown." However, you can tell which object it was by looking at its block ID. The object has a block ID of 2 (### BlockInfo: ID:2), which means it was created second. In the case of this example program, it's trivial to figure out which object was created second. It's not so trivial to locate the client object that has a block ID of 1,233 in a two million–line project. Later, I'll show you a couple of advanced techniques for tracking down a block when all you have is its block ID.

Actions

With actions you can modify TidyHeap's behavior, the AC_CTidyHeapBlock object, or the client object that's being operated on currently. Actions are lightweight subclasses of AC_CTidyHeapAction. TidyHeap's action class architecture makes it easy to customize and add features to TidyHeap. Read on to learn how to install actions, use the actions that ship with TidyHeap, and create your own actions.

Installing into global and object contexts

Action objects are differentiated by their context. There are two types of context: global and object. An action that's executed in a global context operates on all allocated objects, while an action that's executed in an object context operates on only a single object.

TidyHeap manages two different lists of actions that operate in the global context — the global new and global delete actions. When a global new action is installed, it's performed on every block of memory that's processed through

Listing 2. *Leaving undeleted objects*

```
#include "UClientObject.h"
#include "CTidyHeap_AC.h"
void main()
{
  TClientObject* object1 = TH_new TClientObject;
  TClientObject* object2 = new TClientObject;
  TClientObject* object3 = TH_new TClientObject;

  delete object3;
}
```

new; a global delete action is performed on every block of memory processed through **delete.**

For example, AC_CActionVerifyBlock is one of the action classes that ship with TidyHeap. When an AC_CActionVerifyBlock action is installed as a global delete action, every block of memory is verified as it's being deleted. You would install an AC_CActionVerifyBlock action as a global delete action as follows:

```
AC_CActionVerifyBlock* theAction = TH_new
AC_CActionVerifyBlock;
AC_CTidyHeap::Instance()->AcquireDeleteAction(theAction);
```

Here's the code to install AC_CActionVerifyBlock as a global new action:

```
AC_CTidyHeap::Instance()->AcquireNewAction(theAction);
```

All actions, once they're installed, are deleted automatically at the appropriate time by TidyHeap. The method names start with "Acquire" to make it clear that TidyHeap owns the installed action.

When installing an action into an object context, you can install it only as a delete action. You do this by getting the pointer to the client object's AC_CTidyHeapBlock object, and then calling the AC_CTidyHeapBlock class's AcquireDeleteAction method. Once an action is installed in an object context, it will be performed once as the client object is being deleted.

Listing 3 shows how to install an AC_CActionBreakOnDelete action. This action calls DebugStr to break the program into MacsBug when the client object is deleted. AC_CActionBreakOnDelete provides an alternate method for tracking down leaks. If you install this action in a client object that you're suspicious of, and your program never gets the appropriate MacsBug message, you know the client object has leaked. The MacsBug message you would get from the program in Listing 3 when the block is deleted is "Deleting Block 1."

You don't have to use TH_new when creating AC_CTidyHeapAction classes. TidyHeap's objects are tracked differently from client objects so that none of the TidyHeap classes will show up as undeleted blocks.

Built-in action classes

TidyHeap ships with a handful of action classes. You can use some of these action classes in more than one context. For example, as shown in the previous section, you can use AC_CActionVerifyBlock as a global delete action, but there's no reason why you can't also use it as a delete action assigned to a specific object. Just be sure to allocate a new action object for each context you install the action in.

AC_CActionBreakOnDelete and AC_CActionVerifyBlock are described in the previous section. The remaining built-in action classes are described below.

AC_CActionBreakOnNew

In the section "Warnings, Logging, and the Log File" I pointed out that it can be extremely difficult to locate a client object when all you have is the block ID. The action class AC_CActionBreakOnNew helps you do this. You can install an AC_CActionBreakOnNew object as a global new action. AC_CActionBreakOnNew takes a long integer as a parameter in its constructor; this parameter represents a block ID. When the client object with the given ID is allocated with new, the action will break into MacsBug. At this point you can do a stack crawl in your favorite debugger. Here's an example where you're trying to track down the client object with the block ID 1042:

```
AC_CActionBreakOnNew* theAction = TH_new
AC_CActionBreakOnNew(1042);
AC_CTidyHeap::Instance()->AcquireNewAction(theAction);
```

Another way to locate a client object with a block ID is by using the block ID to set a conditional breakpoint in your debugger deep in the guts of TidyHeap. For example, set it on the line in CTidyHeap_AC.cp that increments the counter in the method CreateBlock (currently line 287, blockPtr–>fId = fCount).

Listing 3. *Installing AC_CActionBreakOnDelete*

```
#include "CTidyHeapActions_AC.h"
void main()
{
  TClientObject* myObject = TH_new TClientObject;
  AC_CTidyHeapBlock* objBlock =
    AC_CTidyHeap::Instance()->GetBlock(myObject);
  AC_CActionBreakOnDelete* theAction = TH_new
AC_CActionBreakOnDelete;
  objBlock->AcquireDeleteAction(theAction);
  delete myObject;
}
```

AC_CTidyHeap::CreateBlock allocates the memory for client objects' AC_CTidyHeapBlock objects.

AC_CActionMakeNewFail

The action class AC_CActionMakeNewFail does what its name implies — it takes a long integer n as a constructor parameter that tells the action to make calls to new fail on the nth call. Install an AC_CActionMakeNewFail action as a global new action.

AC_CActionVerifyAll

An AC_CActionVerifyAll action iterates through the list of currently allocated client objects and verifies them all, one at a time. TidyHeap warns you if it finds a corrupt object. Be warned that this significantly slows down your application if installed in a global context. You can use the action class AC_CActionVerifyAll in either of the action contexts.

AC_CActionDoubleDeleteWatcher

The action class is particularly useful. An AC_CActionDoubleDeleteWatcher action marks the client objects sent into **delete** as deleted, but prevents the memory allocated for the client object from actually being deleted. If the object is deleted a second time, TidyHeap will catch it. The downside to installing AC_CActionDoubleDeleteWatcher is that objects being watched are never actually deleted. If you're having problems with a client object being deleted multiple times, I recommend cranking up your application's memory partition and running with AC_CActionDoubleDeleteWatcher installed. It will catch the double deletion as soon as it occurs. You can install an AC_CActionDoubleDeleteWatcher action either as a global delete action or in a specific client object.

Even when an AC_CActionDoubleDeleteWatcher action isn't installed, TidyHeap still looks out for double deletions. Every client object that's deleted is marked, and if it comes through delete again, TidyHeap catches it if possible. The trouble is that in normal operating circumstances, the second time through, the object pointer is pointing at invalid memory, so the "already deleted" mark may not be valid — anything could be occupying that memory. However, in my experience, double deletions usually happen within a very short time of each other, so you have a decent chance of finding the memory still undisturbed. As long as memory is undisturbed, TidyHeap catches the second deletion.

AC_CActionVerifyBlock may catch a double deletion if it's installed as a delete action, because the memory the client object originally occupied may be overwritten, in which case the markers will probably be invalid and you'll get a corrupted-object warning.

AC_CActionExemptBlocks

I added the action AC_CActionExemptBlocks so that it would be possible to tell TidyHeap to ignore certain objects. For example, there are a few objects that MacApp Release 12 creates during startup that are not deleted (this was done to optimize quit times). To ensure that TidyHeap's undeleted-block messages are accurate, you can exempt those blocks. Install the AC_CActionExemptBlocks action as a global new action. Listing 4 shows the code to do this (the code is taken from the sample application Skeleton that ships with MacApp).

AC_CActionGarbageIn and AC_CActionGarbageOut

The AC_CActionGarbageIn and AC_CActionGarbageOut actions fill the client object's memory with garbage values just after allocation and just before deletion, respectively. The values are defined in the objects' class definitions as kGarbageInValue (0xF5) and kGarbageOutValue (0xF6). The values are meant to be recognizable and to cause a bus error if dereferenced. Install AC_CActionGarbageIn as a global new action. You can install AC_CActionGarbageOut either as a global delete action or in a specific client object.

AC_CActionExemptUnknownBlocks

With the AC_CActionExemptUnknownBlocks action installed, TidyHeap only tracks objects created with TH_new. This is handy for times when you have a lot of undeleted blocks and want to debug those created with TH_new separately. I used this action when working with MacApp Release 12, which, as I noted above, deliberately doesn't delete some blocks that were created at startup time. It worked like a charm! Install AC_CActionExemptUnknownBlocks as a global new action.

Listing 4. *Exempting blocks*

```
void main()
{
  // Here's where the client objects we want to exempt are created,
  // so install the action.
  AC_CTidyHeapAction* action = new
AC_CActionExemptBlocks;
  AC_CTidyHeap::Instance()->AcquireNewAction(action);

  InitUMacApp(4);
  InitUPrinting();
  InitUViewSkeleton();

  // Now we want to start tracking objects, so remove the exempter.
  AC_CTidyHeap::Instance()->RemoveNewAction(action);

  // RemoveNewAction doesn't delete the action, so we do it ourselves.
  delete action;

  TApplicationSkeleton* aApplicationSkeleton =
    new TApplicationSkeleton;
  aApplicationSkeleton->IApplicationSkeleton();
  aApplicationSkeleton->Run();
}
```

Listing 5. *AC_CActionBreakOnNew::DoAction*

```
void AC_CActionBreakOnNew::DoAction(AC_CTidyHeapBlock*
inBlock)
{
  if (inBlock->GetID() == fID) {
    char str[80];
    ::sprintf(str, "AC_CActionBreakOnNew: Block ID: %ld",
fID);
    ::c2pstr(str);

    ::DebugStr((StringPtr) str);
  }
}
```

AC_CActionIncrementalNewFailer

The AC_CActionIncrementalNewFailer action is specially designed for you QA folks to make your engineers' lives miserable. AC_CActionIncrementalNewFailer makes new fail sequentially — first new number one fails, then number two fails, then number three, and so on. The kicker is that TidyHeap saves the count into a text file so that the next new that will fail is persistent between runs of your application. This stresses your failure-handling code to the limit! Install this action as a global new action.

Rolling your own action classes

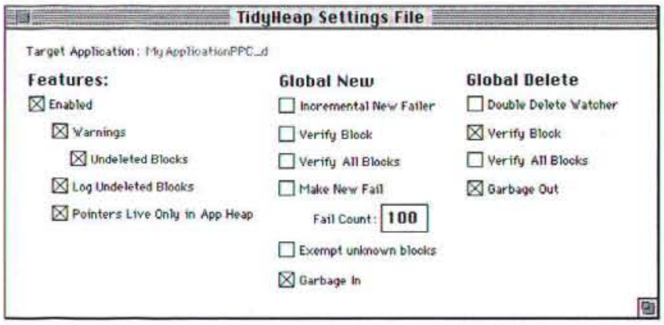
The AC_CTidyHeapAction class is a very simple and lightweight class. To make your own action class, simply subclass AC_CTidyHeapAction and override the pure virtual function DoAction. The function takes as a parameter a pointer to an AC_CTidyHeapBlock object, which is the object TidyHeap is currently operating on. Listing 5 shows the code overriding DoAction for the AC_CActionBreakOnNew action class.

Changing Settings with TidyHeap Director

TidyHeap writes its current settings to a file named *myapplication.prf* in your application's folder. It saves to this file any changes to TidyHeap settings made between runs of your application so that you can use the TidyHeap Director tool to edit the file and change the behavior of TidyHeap without recompiling your code.

Figure 2 shows the TidyHeap Settings File dialog box. The application you're working with appears after "Target Application." On the left are five Features preferences.

- Check Enabled to turn TidyHeap on; unchecking this option effectively turns off TidyHeap completely.
- Check Warnings to send warnings to MacsBug.


TidyHeap Settings File

Target Application: MyApplicationPPC_d

Features:
- ☒ Enabled
  - ☒ Warnings
    - ☒ Undeleted Blocks
  - ☒ Log Undeleted Blocks
  - ☒ Pointers Live Only in App Heap

Global New
- ☐ Incremental New Failer
- ☐ Verify Block
- ☐ Verify All Blocks
- ☐ Make New Fail
  - Fail Count: 100
- ☐ Exempt unknown blocks
- ☒ Garbage In

Global Delete
- ☐ Double Delete Watcher
- ☒ Verify Block
- ☐ Verify All Blocks
- ☒ Garbage Out

Figure 2. *Setting TidyHeap preferences with TidyHeap Director.*

- If Warnings is enabled, you can set Undeleted Blocks, which causes TidyHeap to send a list of undeleted blocks to MacsBug when your application terminates.
- Check Log Undeleted Blocks to enable logging of undeleted blocks to the log file (see the section "Using the Log File"). If you have many undeleted blocks, the logging process can take a while and you may want to shut off the logging.
- Check Pointers Live Only in App Heap to tell TidyHeap to verify that pointers you send into GetBlock or VerifyBlock are within your application's memory heap.

Use the checkboxes under Global New and Global Delete to automatically install the listed actions as global new or delete actions when TidyHeap starts up.

The settings file is read once — when TidyHeap is created. To make changes to your settings, make sure your application isn't running at the time. The next time your application is run, your new settings will be loaded.

Alternative to TidyHeap

There are other tools with features similar to TidyHeap. This section tells you a little about some of them and compares them with TidyHeap.

Metrowerks ships a tool called DebugNew. You'll find the library and a demo for DebugNew on the CodeWarrior CDs. DebugNew has some of the same features as TidyHeap, but it's not as extensible and doesn't have as many features. The features it does have are quite valuable, however: it can check for leaks, validate blocks of allocated memory, and verify that a block is in the application heap. DebugNew also has a macro like TH_new called NEW, and it has a log file. DebugNew is smaller than TidyHeap and is written in C, so it may run in a smaller footprint and it doesn't require C++.

Another tool for tracking memory leaks is the ZoneRanger application, also from Metrowerks. ZoneRanger is quite good not only at tracking memory leaks, but also at fine-tuning your application's heap and managing memory. This is a great tool that every programmer should have at hand.

QC and Spotlight by Onyx Technology are MacOS system extensions that provide memory debugging for applications without requiring that you modify the source code or recompile the application. QC stress tests applications for all kinds of memory-related errors. These tools work well with applications that directly allocate memory with the Macintosh Memory Manager or that manage resources returned by the Resource Manager, and can be used in conjunction with TidyHeap.

Memory Mine by Adianta is a standalone tool for monitoring heaps and stress testing applications. It flags heap corruption when it happens and you can see memory leaks as they occur. Source code is not needed to monitor heaps.

Apple ships a handful of dcmds (debugger commands for MacsBug) that are extremely helpful as well: Blat makes sure that nobody writes to the first 256 bytes of low memory, DisposeResource catches resource handles that are passed to DisposeHandle, Leaks watches for memory leaks when you tell it to (via "leaks on"/"leaks off" commands to MacsBug), EvenBetterBusError watches for dereferences of NIL and writes to NIL (location 0), and Xap fills pointers and handles that were disposed of with a value that causes a bus error if dereferenced. Furthermore, the useful system extension DoubleTrouble catches double disposes of a handle.

Another very useful tool from Apple is the Debugging Modern Memory Manager, which, not surprisingly, is a debugging version of the Modern Memory Manager. It performs many of the functions described in this article, including filling a client object's memory with garbage, checking for writes past the end of blocks, and checking for bad handles passed into Memory Manager routines.

Terminate

Well, there you have it. You've got TidyHeap hooked into your framework or application and you're ready to let it track your client objects and help you figure out where a leaky block was created, using either the TH_new macro or its block ID. You can now squash all your memory leaks with ease. You also learned about some of the more advanced features of TidyHeap, including the action classes that ship with it. And, you're ready to use TidyHeap Director to modify TidyHeap's behavior without recompiling your application.

I hope that TidyHeap makes it possible for you to focus on making insanely great software without having to worry so much about the more irritating and mundane tasks of application programming, such as tracking undeleted objects. Happy block hunting!

Thanks to our technical reviewers Brian Arnold, Paul Black, Geoff Clapp, Mike Rossetti, and Jason Yeo.

develop

by Dave Evans

Code Mechanic: Better Than Ever Stress Testing

Dave Evans

There are few things more frustrating than losing access to your debugging tools due to a freeze, because you can't fix what you can't diagnose. The best course is to stop freezes before they start, so I'd like to share a common cause of freezes I've found. I'll also discuss some of the stress-testing options that are available to help you catch freeze-causing problems you might have missed, including an improved debugging tool.

Veteran readers of develop may notice a new title for this column. The previous title, "Balance of Power," was apt for its time, indicating a focus on PowerPC issues. But now that all new MacOS computers are PowerPC-based, everybody's writing about PowerPC, and my efforts in this area are complete. This new title reflects a focus on the mechanics of code tuning, with tips for improving your application's performance and stability, which I hope you'll find just as useful.

PROTECT YOUR VECTORS

Even if you use a PowerPC-based MacOS computer, the first 256 bytes of memory are dedicated to 680x0 exception vectors, which the 680x0 software emulator uses to emulate 680x0 exceptions and interrupts. On a 680x0-based computer, these values are read by the processor itself when handling an exception or servicing an interrupt.

Under System 7, these important vectors are not memory protected. Any program can read from or write to them, possibly resulting in a serious failure. While not all of the vectors are used, modifying some of them will cause an immediate freeze, leaving you without access to your debugging tools. You probably don't address these vectors intentionally, but it often occurs accidentally when a nil pointer or empty handle is de-referenced.

Unintentionally reading from these vectors will produce a random result. In most cases the vectors are addresses of special system routines; these vectors can have any value, and they vary significantly from one computer model to another. As an example of how easy it is to cause a problem in this area, take a look at the following C code, similar to that found in some applications:

```
front_window = FrontWindow();
if (front_window->windowKind < 0)
  MyDeskAccessoryRoutine(front_window);
```

The developers didn't realize that FrontWindow can return nil when no windows are open. In that case the application de-references the nil pointer and makes a logical decision based on the sign of the half word at $6C in low memory, which is the high half of the interrupt level 3 vector. On most Macintosh computers released before 1995, this vector pointed into ROM starting at address $40800000. Because of this, the applications would test the high half word value of $4080, and they wouldn't run the desk accessory routine. This was the right behavior, but for the wrong reason; disaster was averted by luck.

Beginning with all PCI-based PowerPC computers, ROM starts at location $FFC00000. During the development of these computers, we found that applications with code like the above would crash because they executed unexpected code after comparing the new half word value of $FFC0. We were able to work around their problem by changing the interrupt level 3 vector to point to a routine in RAM. This changed the high half

Dave Evans is a typical American breed, often found hard at work on the Apple campus. He lacks a pure pedigree, coming instead from Irish, Norwegian, Scottish, and Hungarian stock. This mix gives him an even temperament and friendly demeanor. He makes a loving companion, good for playing catch or Frisbee, and he doesn't eat much. If you have a large yard, please contact your local animal shelter or kennel for adoption details.

word value to be a small positive number, and the applications behaved as expected. Still, the best case would have been if the problem could have been avoided in the first place. The following code is an example of what would have been a better, crash-free approach:

```
front_window = FrontWindow();
if (front_window && front_window->windowKind < 0)
  MyDeskAccessoryRoutine(front_window);
```

Checking for nil pointers or handles is one way you can avoid these crashes in the first place. Checking for empty handles is another necessary step, since unlocked relocatable blocks that are marked purgeable may disappear any time memory can move.

To detect problems with purgeable blocks, you'll need tools to stress test your application. Utilities that display heap zones, allowing you to compact and purge a heap on demand, are a good start. For serious testing, however, you'll need a stress tool that operates all the time. One good tool for this is MemHell, which will compact and purge your heap whenever a Memory Manager routine that might move or purge memory is called. This slows down execution of your tests, but it will flush out problems with purgeable blocks.

So, while accidentally reading from low memory can cause unexpected results, accidentally writing to low memory can be fatal, and this is one of the most common causes of freezes that I've noticed. You may think this could never happen in your code, because none of your blocks are purgeable and you always check errors after allocating pointers. Think again; there are plenty of other opportunities. Do you check for an error after every GetResource call? Getting an unexpected error — from a corrupted resource file, for example — is one way you can end up with a nil handle. Besides diligent review of your code, you need to do stress testing to flush out possible errors, or freezes are likely to result.

Are You Stressed Enough?

There are a number of tools to help add stress to your testing. I've already mentioned MemHell for finding problems with purgeable blocks. You'll similarly need a tool to find reads and writes to the exception vectors.

The simplest choice is the ubiquitous and venerable EvenBetterBusError, written by Greg Marriott. This tool safeguards the first four bytes of memory, which are very often accidentally written over or read from. To detect reads, it places in the first four bytes of memory a value which when de-referenced will cause a crash. If you use a nil pointer or empty handle, the illegal value is likely to be used as data or de-referenced, leading to a crash. To detect writes, it checks periodically to see if the value that it placed has been overwritten; if so, you'll be notified with a DebugStr message. EvenBetterBusError is included as a dcmd in MacsBug beginning with version 6.5.4.

I've extended EvenBetterBusError to be more aggressive. The new version, YetEvenBetterBusError, writes a value over the first 256 bytes of memory which will cause a crash into your debugger when de-referenced. It also checks periodically for writes to these locations, but more frequently than EvenBetterBusError does. Like EvenBetterBusError, upon noticing a write to these locations it will notify you with a DebugStr message. YetEvenBetterBusError can be found at <http://www.mactech.com>.

To implement YetEvenBetterBusError, I had to sacrifice some compatibility with existing applications. Any application code that assumes the exception vectors start at address 0 will no longer function correctly. Most applications don't use the exception vectors directly, but some copy protection schemes do modify the vectors.

The correct way to determine the location of the exception vectors is by using the 680x0 instruction MOVEC, which must always be executed in supervisor mode. The location of the first vector is stored in the 680x0 VBR (Vector Base Register). To read the address, you would write the following assembly code:

```
_EnterSupervisorMode  ; old sr result in d0
movec    vbr,a0       ; get the vbr
move.w d0,sr          ; restore the old sr
```

Always use the VBR to find these vectors. Although early versions of the MacOS always placed them at location 0, they're now often elsewhere. When virtual memory is turned on, for example, the vectors will actually reside in the system heap, and the VBR will point to them. To maintain compatibility, however, if virtual memory doesn't handle an exception it calls through to the original vector table at location 0. This is why even with virtual memory on, writing over the low-memory exception vectors can still cause a freeze.

YetEvenBetterBusError is able to overwrite and then monitor the first 256 bytes of memory by moving the exception vector table entirely. So, even when virtual memory is on, with YetEvenBetterBusError installed the original low-memory vectors are never called. This is why some existing applications may be incompatible with YetEvenBetterBusError.

A Cure for Test Anxiety

It's true that fully testing your code to reflect all possible configurations and user actions can be a near-impossible task. But the perceived stability of both your application and the computer depends on how well we all write and test our software. To do the best possible job, use the stress-testing tools mentioned in this column or in the article "Squashing Memory Leaks with TidyHeap" in this issue. Do the right thing: stress test, then relax!

Thanks to Pete Gontier, Chris Jalbert, Bo3b Johnson, Dave Lyons, Quinn "The Eskimo!", and Keith Stattenfield for reviewing this column.

develop

by Apple Developer Support Center

Macintosh Q & A

Q: How can I determine whether my 680x0 program is running on a PowerPC machine?

A: The best way to do this is to use the Gestalt selectors as described in the Universal Interface file Gestalt.h:

```
enum {
 gestaltSysArchitecture = 'sysa',  // Native system architecture
 gestalt68k = 1,                   // Motorola MC680x0 architecture
 gestaltPowerPC = 2                // IBM PowerPC architecture
};
```

A snippet of code that uses this would look something like the following in your 680x0 program:

```
long myattr;
OSErr  err;

 err = Gestalt( gestaltSysArchitecture, &myattr);
 if (err == noErr) {
  if (myattr == gestaltPowerPC)
    // Do some PowerPC thing.
    ...
 } else {
    // Handle the error condition.
   ...
 }
```

Q: I've created a dialog box that has two editable text fields in it and I've used an 'ictb' resource to change the font and the size of the fields to 10-point Geneva. When I move the insertion point into the second field and delete a character, the character just before the one I deleted drops down a couple of pixels and overwrites itself. What can be done about this?

A: You've run into a problem with the Dialog Manager's support of 'ictb' resources. The Dialog Manager forgets to reset some of the fields of the TextEdit record when it swaps the font and font size information stored in the 'ictb'. You need to reset the fontAscent and lineHeight fields of the TextEdit record to match the size of the font specified in the 'ictb'. By default those fields are set to the lineHeight and fontAscent of 12-point Chicago.

The following code shows how to set up the TextEdit record properly.

```
static void SetUpEditField(DialogRef dlog, short fontNum,
short fontSize)
{
 FontInfo   info;
 DialogPeek dpeek = (DialogPeek)dlog;

 if (dpeek != nil) {
   TEHandle te = dpeek->textH;              // Get the TEHandle.
   if (te != nil) {
     short  oldFont = dlog->txFont;         // Save old font info.
     short  oldSize = dlog->txSize;
     TextFont(fontNum);                     // Set correct font info.
     TextSize(fontSize);
     GetFontInfo(&info);
     // Fix the TE record.
     te[0]->txFont = fontNum;               // Set font.
     te[0]->txSize = fontSize;
     // Calculate the correct line height.
     te[0]->lineHeight = info.ascent + info.descent +
info.leading;
     te[0]->fontAscent = info.ascent;
     TextFont(oldFont);                     // Reset the font info.
     TextSize(oldSize);
   }
 }
}
```

Then call SetUpEditField in your dialog routine, like this:

```
static short DoDialog(short resID)
 {
  DialogRef  dlog;
  GrafPtr    oldPort;
  short      itemHit = 0;

   GetPort(&oldPort);
   dlog = GetNewDialog(resID, nil, (WindowRef)-1);
   if (dlog) {
     SetPort(dlog);
     SelectDialogItemText(dlog, 2, 0x8000, 0x8000);
                                   // Set the cursor.
     SetUpEditField(dlog, geneva, 10);
                                   // Set the edit field.
   (void)SetDialogDefaultItem(dlog, 1);   // Highlight OK button.
     ShowWindow(dlog);             // Show the dialog.
     while (itemHit != ok)
       ModalDialog(nil, &itemHit);
     SetPort(oldPort);
     DisposeDialog(dlog);
   }
   return itemHit;
 }
```

Of course, an alternative to all this is to avoid the Dialog Manager altogether and thereby avoid these obscure Dialog Manager problems. MT

Developer DEPOT™

- Web site: http://www.devdepot.com • E-mail: orders@devdepot.com
- Phone: 1-800-MACDEV-1 (Outside the U.S. & Canada: 805-494-9797)
- Fax: 805-494-9798

✓ **Hundreds of developer products**

✓ **Satisfaction and lowest price guaranteed**

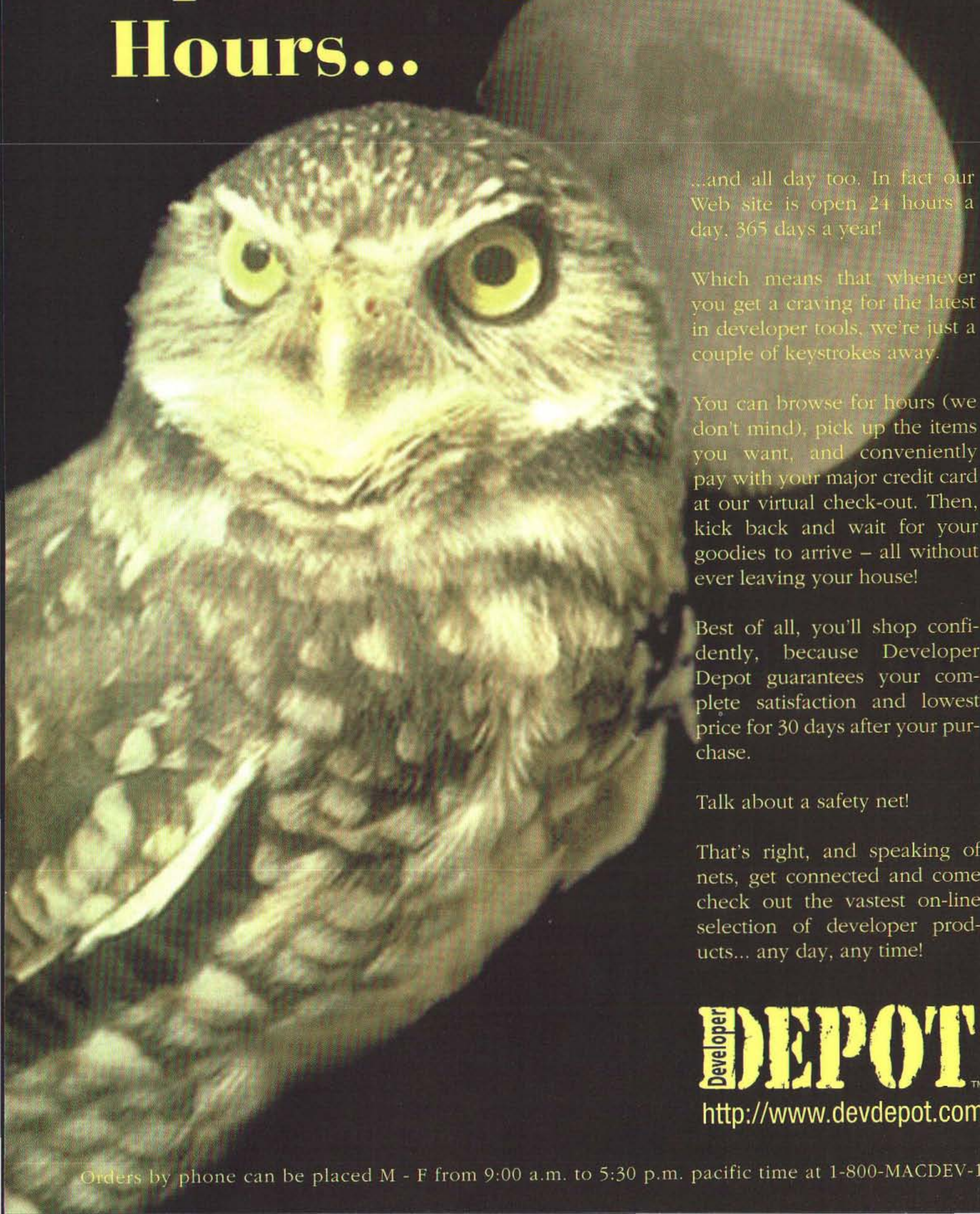
✓ **World renowned customer service**

✓ **Order by phone, E-mail, fax or through our continually updated Web site**

NOW CARRYING APPLE PRODUCTS!

New Categories!
- ***Multimedia***
- ***Games***

Open Programmer's Hours...



...and all day too. In fact our Web site is open 24 hours a day, 365 days a year!

Which means that whenever you get a craving for the latest in developer tools, we're just a couple of keystrokes away.

You can browse for hours (we don't mind), pick up the items you want, and conveniently pay with your major credit card at our virtual check-out. Then, kick back and wait for your goodies to arrive – all without ever leaving your house!

Best of all, you'll shop confidently, because Developer Depot guarantees your complete satisfaction and lowest price for 30 days after your purchase.

Talk about a safety net!

That's right, and speaking of nets, get connected and come check out the vastest on-line selection of developer products... any day, any time!

Developer DEPOT™

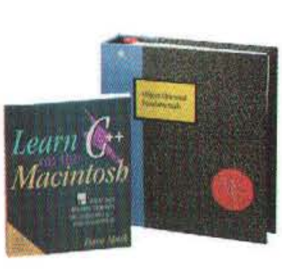
http://www.devdepot.com

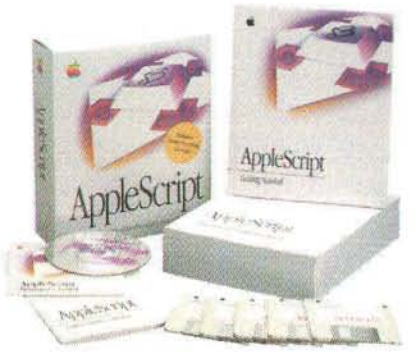
Orders by phone can be placed M - F from 9:00 a.m. to 5:30 p.m. pacific time at 1-800-MACDEV-1.

ANNOUNCEMENT

Developer DEPOT™

now carries Apple brand products





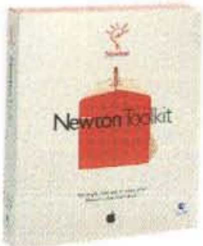


TABLE OF CONTENTS

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

One & 2-Day SHIPPING (call for details)

Order Toll-free 800-MACDEV-1 (8006223381)

Developer Depot 30 day Money Back, Price and Satisfaction Guarantee

Developer Depot products are sold with a 30 Day money back guarantee on user satisfaction, lower prices and against defects. If, for any reason, you are not satisfied or find the same product at a lower price within 30 days, please call Customer Service at 800-MACDEV-1 and request a Return Merchandise Authorization (RMA) number to get a full refund or the difference in price (where applicable). You must return undamaged product at your expense, including all its original packaging, documentation and the blank warranty card if applicable. Developer Depot will replace defective product upon receipt of the defective merchandise. Please remember to back up your data before installation of any new hardware, software, or peripherals; we cannot be responsible for any lost data. Policies, item availability, and prices are subject to change without notice. The price in effect when we receive your order will be the price that you are charged. We are not responsible for any typographical errors in this or any other catalog, nor for any misstatements from any vendor. Purchase orders are also accepted but must be in writing, signed, come with a contact person and a telephone number, and mailed to P.O. Box 5200, Westlake Village, CA 91359-5200. Faxed copies and purchase order numbers alone are not acceptable.

Stuff our lawyer made us write.
Developer Depot makes no other warranties. All other warranties, either expressed or implied, including the implied warranty of merchantability and fitness for a particular purpose are disclaimed. Developer Depot shall not be liable for any direct, special, incidental or consequential damages including lost profits, from any delay in delivery, or for any personal injury arising from the use of any product sold through Developer Depot. The limit of direct damages, if any, shall not exceed the purchase price of the product.© 1997 Xplain Corporation. All rights reserved. Any unauthorized duplication is in violation of federal laws. Developer Depot is a registered trademark of Xplain Corporation. All product names in this catalog are the trademarks of their respective holders.

For Macintosh Programmers & Developers

MacTech™ MAGAZINE

Subscriptions: US magazine: **$47** for 12 issues (MTYRDM)
Canadian: **$59** for 12 issues (MTYRCM)
International: **$97** for 12 issues (MTYRFM)
Back Issues: **$10** each (subject to availability)

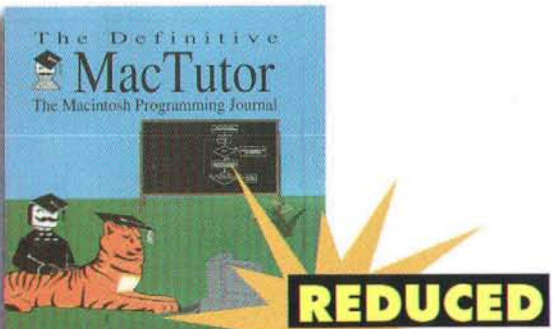
Limited time offer: Buy **Volume 11** ***NOW***, and get a **free** upgrade to the soon-to-be-released **Volume 12!** (Free shipping on the upgrade!)

MacTech CD-ROM Volumes 1-11

- 1420+ articles, from all 127 issues of MacTech Magazine (1984-1995)
- Improved hypertext, and a new THINK Reference Viewer - for lightning quick access! New hyperlinks between articles
- 100+ MB of source code - use them in your applications, with no royalties!
- Full version of THINK Reference™ – the original online guide to Inside Macintosh, Vols. I-VI
- 80MB of FrameWorks/SFA archives and the most complete set of FrameWorks archives known
- Sprocket™! MacTech's Tiny Framework that compiles quickly and supports System 7.5 features
- The best threads from the Macintosh programmer newsgroups plus thousands of notes, tips, snippets, and gotchas
- Popular tools that Macintosh programmers use to increase their productivity and much more!

List Price $89 Our Price **$79** (SMTCD11)

Upgrades: Our Price **$39** (SMTCD11U)



The Best Of MacTutor Volume 4

List Price $29.99 **Our Price $4.99** (BBEST4)

MacTech Mouse Pad

Slide on this! With an extra-large surface (11" by 10") and a deluxe sleek plastic coating, you'll be zooming across your screen in no time at all. Speed limit not enforced!

Our Price **$8.95** (AMTPAD)

System 7.5 Technologies

by Apple Computer, Inc.

- Self-paced course designed to allow software developers to write code that extends the functionality of an application for System 7.5.
- Contains comprehensive materials for drag-and-drop, threads, standard mail package, and QuickDraw GX printing.

Students should be familiar with the basics of developing an application on the Macintosh. Metrowerks CodeWarrior Lite is included on the CD with the lab exercises. The lab assignments were developed in CodeWarrior 8. The labs can also be done in another development environment but project files for them are not provided.

Training Format: Tutorial with labs.

Requirements: Macintosh or Mac-OS compatible computer with a 68020 processor or greater (PowerPC preferred); 8 MB RAM; 25 MB hard disk space; System 7.5 or later; CD-ROM drive.

Our Price **$49.95** (SSYSTECH)

Apple Guide Integration

by Apple Computer, Inc.

- Self-paced overview teaches you when and how to add Apple Guide help to your program.
- Powerful help system that can guide the user through a task.
- Tutorial will lead you through the steps necessary to integrate Apple Guide into your application. CodeWarrior Lite is included on the CD with the lab exercises.

Training Format: Overview with labs.

Requirements: Macintosh or Mac-OS compatible computer with 68020 processor or greater, PowerPC preferred; 8 MB RAM; 25 MB hard disk; System 7.5 or later; CD ROM drive.

Our Price **$49.95** (SAGI)

Virtual Tutor for QuickTime VR

By Apple Computer, Inc.

- Self-paced, hands-on course, which provides a comprehensive environment for learning the steps of the QTVR development process. The student can cover all of the topics or choose areas to focus on. Topics covered include: QTVR capabilities and key concepts, panoramic movies, object movies, QTVR Scene movies and authoring with QTVR.
- CD-ROM contains lots of useful examples and demos. In addition to all the step-by-step exercise files.

If the student completes the entire course, he/she will create a complete, authored multimedia project similar to the demonstration title that comes on the enclosed CD-ROM. There are approximately 3–4 days of training.

Training Format: Tutorial with labs.

Requirements: 40 MB RAM minimum, 64 MB preferred; Macintosh or Mac OS-compatible computer with a 33 MHz 68040 processoror greater; System 7.1 or later; CD-ROM drive; 17"color monitor.

List Price **$79.95** (SVTFQTVR)

Newton

Newton Toolkit 1.6

With Newton Toolkit, you can easily create software that runs on any Newton PDA, including Apple's MessagePad and Motorola's Marco.

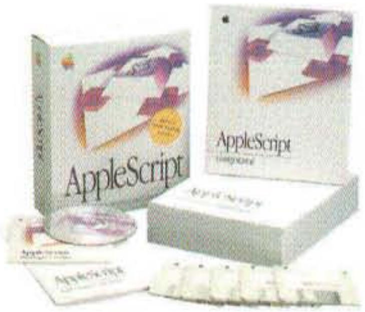
- Now supports Newton 2.0
- Dynamic Language Eases Development
- Allows you develop applications interactively
- New Compiler Enables Faster Applications

Newton Toolkit 1.6

Our Price **$299** (SNETO)

Newton Toolkit Update 1.6

Our Price **$49** (SNETOUP)



AppleScript Software Development Toolkit 1.1

- AppleScript language, system software extension, and script editor
- FaceSpan 1.0
- Developer's redistribution license for AppleScript System software extension and FaceSpan runtime code

Our Price **$49** (SASDT)

Newton Programmer's Guide for Newton 2.0

- The Newton Programmer's Guide consists of two volumes covering the Newton System Software, and one volume covering Newton Communications
- The two-volume set, Newton Programmer's Guide: System Software, is the definitive guide and reference for Newton programming. This set of volumes explains how to write Newton programs and describes the system software routines that you can use to do so
- The Newton Programmer's Guide: Communications, describes the Newton communications system software for version 2.0

Our Price **$149** (BNPGFN)

Apple Dylan

dynamic object oriented language and development environment

Apple Dylan Technology Release

- Contains a PowerPC-native prototype version of a development environment based on the Object Oriented Dynamic Language (OODL) Dylan. Developers will be able to produce code targeting both 680x0 and Power Macintosh systems
- Automatic memory management
- Application framework and user-interface &builder
- High-level exception handling
- Cross-language support for C code and APIs

CD & Online Documentation Our Price **$39.95** (SADTRO)

CD & Hardcopy Our Price **$59.95** (SADTRH)

Apple Media Tool Programming Environment 2.0

- This object-oriented language and application framework allows programmers to customize features used within the Apple Media Tool authoring environment
- Includes an expanded Apple Media Language (AML) class library, incremental compiling and linking of AML code, faster debugging facilities, Macintosh Programmers' Workshop (MPW), and user-oriented documentation written from an AMTPE developer's perspective
- Portable across 68K, Power Macintosh, and Windows platforms

Our Price **$995** (SAMTPE)

Multimedia Authoring with Apple Media Tool

by Apple Computer, Inc.

Apple Media Tool offers new multimedia users a way to get started creating interactive multimedia with minimal learning time. This self-paced tutorial will make Apple Media Tool (AMT) even easier to understand and to use. Using this tutorial, you will create a realistic multimedia project using exciting techniques such as QuickTime movies, animation and more. A demo version of AMT is included and can be used for the exercises. Training Format: Tutorial with labs.

Our Price **$49.95** (SMWAMT)

Virtual Reality Programming with QuickTime VR 2.0

by Apple Computer, Inc.

- Virtual Reality Programming Book/CD-ROM for QuickTime VR 2.0
- Enables you to write C and C++ programs using QuickTime VR 2.0
- Allows QuickTime VR to be used in games, multimedia titles and other programs
- QuickTime VR 2.0 objects can be zoomed in on, panned, or linked with hots spots
- Both panoramas and objects have hot spots linked to World Wide Web URLs

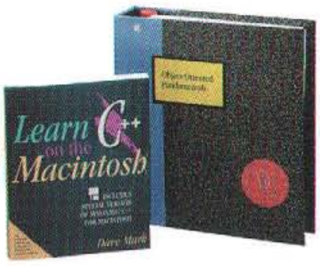
Our Price **$49** (SVRPQT)

QuickTime Developer's Kit 2.0

by Apple Computer, Inc.

- QuickTime 2.0 Extension, QuickTime Power Macintosh Extension, and QuickTime Musical Instruments extension
- Utilities like MoviePlayer 2.0, 16-bit Audio Compression, etc.
- Sample content such as MPEG Movies, Music Movies, Time-Code Movies, and 60 field per second movies
- Includes software-only playback features such as faster 2x playback mode for current compressors, Apple Cinepak compressor, 1-bit fast dithering, network tuning, load-into-RAM option, and Photo CD support

Our Price **$99** (SQTDK)



Object-Oriented Fundamentals 1.1

by Apple Computer, Inc.

- DU's multimedia Object-Oriented Fundamentals course enables you to easily make the paradigm shift from procedural to object-oriented design. It will introduce you to the entire object skill set, from general concepts through analysis and design
- This course includes a "lite" version of Metrowerks CodeWarrior integrated development environment for use in the labs. This course also includes a copy of the book Learn C++ on the Macintosh by Dave Mark

Our Price **$245** (SOBORFU)

QuickTime VR 2.0 Authoring Tools Suite

by Apple Computer, Inc.

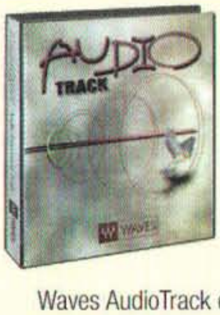
- QuickTime VR is a cross-platform software from Apple which enables webpage designers and professional developers to create new multimedia products and webpages incorporating QuickTime VR content. With QuickTime VR, users interactively navigate through 360° views of space, and explore three dimensional objects on Macintosh or Windows-based personal computers
- The QuickTime VR Authoring Tools Suite is a set of Macintosh tools to create and link panoramas and objects from photographic, digital, video, or computer generated images
- Included is a complete set of documentation for planning, designing, photographing, and creating QuickTime VR panoramas and objects. The authoring tools also allow you to link objects to panoramas using clickable hot spots

Included on the CDs are:

- A software tool (MPW-based) that stitches and blends adjacent images into a panoramic PICT file
- A software tool (MPW-based) that dices and compresses panoramic PICT files to less than 100 KB (low resolution) per panorama
- A scene editor (HyperCard-based) to create QuickTime VR scenes by adding and positioning nodes, hot spots, linking nodes together, and for linking QuickTime VR objects to scenes
- A variety of utility tools for formatting the data into the runtime software

Due to a revolutionary distortion-correcting algorithm, QuickTime VR panoramas and objects maintain a normal perspective when the user moves the mouse. The speed of the algorithm allows up to 24-bit color images. Both vertical and horizontal panning can occur at fast speeds.

Our Price **$395** (SQTVRATS)



AudioTrack
by WAVES

AudioTrack is a software plug-in for native processing on digital audio recording and editing systems.

Waves AudioTrack combines the most-needed audio processors into a single piece of software, including 4 bands of equalization, compression/expansion, and noise gating. AudioTrack is ideal for preparing audio for InterNet streaming formats, processing individual mono/stereo tracks of audio and audio for video editing systems including digital sequencers.

Feature list
- A single window interface
- 4-band ParaGraphic Equalizer, Compressor/Expander and Gate
- Instantaneous A/B comparisons of on-line settings
- Pretested setup libraries supplied for various processing solutions
- Power Macintosh native processing (requiring no DSP board)
- Volume and gain reduction meters
- Peak hold and clip meters

Requirements:
The AudioTrack is compatible with all machines supported by Deck II, SoundEdit 16, Adobe Premiere 4.0 and Cubase VST 3.1.

Our Price **$270** (SAUDIOTRK)

Order Toll-free
800-MACDEV-1
(800-622-3381)

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

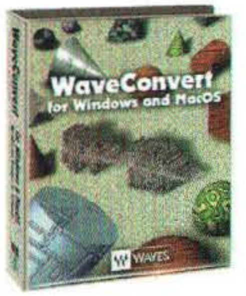
Native PowerPack
by WAVES

Waves Native PowerPack - a complete pro audio system of Waves software that utilizes native processing within the computer, requiring no additional hardware. The Waves Native PowerPack is a complete effects processing solution for those making records, creating multimedia titles, authoring audio for the InterNet, designing sounds for games, or recording at home.

The Native PowerPack is available on the Macintosh for Macromedia SoundEdit16 and Deck II, Adobe Premiere, Bias Peak, Steinberg Cubase VST and Digidesign Audiosuite, and on Windows95 and WindowsNT using Microsoft ActiveMovie architecture to support Sonic Foundry Sound Forge , Steinberg WaveLab, Twelve Tone Systems CakeWalk, and other platforms that are using Microsoft ActiveMovie architecture.

Native Power Pack for Mac or PC includes: L1 - Ultramaximizer peak limiter, Q10 - Paragraphic EQ, C1 - Compressor/gate, S1 - Stereo Imager, TrueVerb -virtual/space reverb, and WaveConvert - Batch converter.

Our Price **$600** (SNPWRPCK)



WaveConvert
by WAVES

WaveConvert is an easy to use cross-platform audio processing application for multimedia developers. It's ideal for all multimedia audio productions, delivering loud and clear audio files. It combines the well-known peak limiting technology of the Waves L1 Ultramaximizer with other Waves processors into a powerful audio processing tool which offers one step conversion of multiple files.

- Sample rate conversions from 5kHz to 48kHz
- Audio level normalization
- Preview function
- Stereo/mono conversion
- Tone and gain control
- Reduced background noise
- 16/8 bit conversion using special audio maximization techniques
- Preparation for RealAudio
- AIFF, ".snd", .wav, SDII, and RAW formats

Our Price **$270** (SWAVECON)

Clip VR™

Clip VR™
by eVox Productions

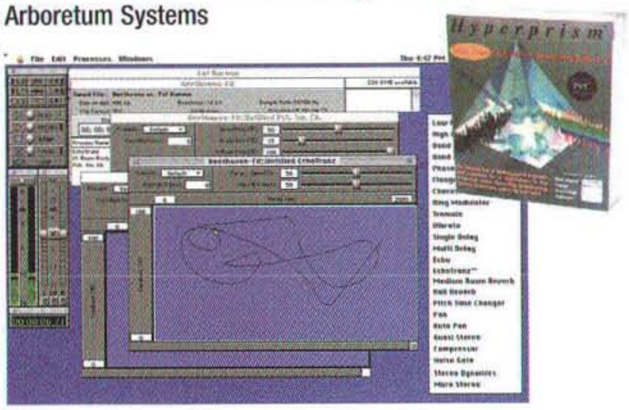
Clip VR™ is a new digital image library offering high quality Photographic Virtual Reality (PVR) images for use with Quicktime® VR and other desktop VR tools.

Clip VR™ Panoramic Image components include alpha channel masks. Combine elements into a composite panorama which is converted to the finished QuickTime VR movie using Make QTVR Panorama Tool. The Components ("Clips") include complete 360 degree scenes, libraries of 360 degree terrains, 360 degree skies, buildings, and objects. Images are provided as .PICT files AND QuickTime VR movie files. Clip VR™ allows you to create high quality VR worlds from pre-photographed component images. Clip VR™ can be used to add excitement to a web site, to increase the interactive value of a CD-ROM, or simply for fun! Requires imaging editing program such as Adobe Photoshop™ to perform .PICT image compositing.

Our price **$89.95** (SCLIPVR)

Hyperprism-PPC Real-Time Sound Effects Processing Software

by Arboretum Systems



Hyperprism-PPC is highly flexible and creative effects processing software for the PowerMac. Its unique Blue Window™ interface allows for dynamic, on-the-fly, "analog-like" gestural control over multiple effects processing parameters simultaneously.

- 24 high quality real-time processing algorithms
- Non-destructive, dynamic, multipass processing
- Presets provide ready-to-use settings for instant results
- No additional audio hardware is required
- I/O is also possible with 3rd party sound cards
- Supports mono/stereo Sound Designer II/AIFF files, 8/16 bits at standard sample rates

Requirements: Power Macintosh or Mac OS-compatible with PowerPC processor; 8 Mb RAM; 1 Mb HD space; System 7.1 or later; Sound Manager 3.1 or later recommended.

List Price **$300** (SHYPERPPC)

Hyperprism-MMP (Multi Media Producer) Real-Time Sound Effects Processing Software

by Arboretum Systems

Hyperprism-MMP is a package of highly flexible and creative effects processing plug-ins for applications supporting the Adobe Premiere plug-in architecture.

Requirements: Power Macintosh or Mac OS-compatible with PowerPC processor; Adobe Premiere-compatible host application; 3mb RAM; 1mb HD space; System 7.1 or later; Sound Manager 3.1 or later recommended.

List Price **$250** (SHYPERMMP)

Hyperprism-TDM Real-Time Sound Effects Processing Software

by Arboretum Systems

Real-time sound effects processing software dedicated to the Digidesign TDM NuBus platform. Hyperprism-TDM is considered as the only set of Plug-Ins on the market to cover most of the needs of professional recording studios and post production houses.ce.

List Price **$945** (SHYPERTDM)

Hyperprism-68K Real-Time Sound Effects Processing Software

by Arboretum Systems

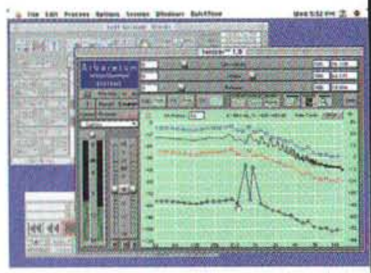
Hyperprism is a real-time sound effects processing software for Audio Media I&II, Sound Tools I&II, or Pro Tools 442 NuBus systems. It takes full advantage of the 56K DSP (Digital Signal Processor) that resides on these audio cards and works great on faster Macs such as Quadras.

Requirements: Any Macintosh running System 7.0 or higher with NuBus slots and either a Sound Tool I&II, or Audio Media I&II, or Pro Tools 442 systems; 8 Mb of Ram; 6 Mb of HD space.

List Price **$389** (SHYPER68K)

Ionizer Real-Time Dynamic Spectral Reshaping Tool

by Arboretum Systems



Ionizer is a real-time 512-band gated equalizer and multiband dynamics processor. Typical applications for Ionizer include broadband noise reduction, audio restoration and repair, mastering, multimedia, broadcasting, sound design, special effects, and anywhere you need powerful EQ and dynamics control.

Requirements: Power Macintosh or Mac OS-compatible with PowerPC processor; 8 Mb RAM; 8 Mb HD space; System 7.1 or later; Sound Manager 3.2.1 or later recommended.

List Price **$800** (SIONIZER)

Captivate 4.6: Essential Graphics Utilities

by Mainstay

Captivate™ 4.6 is a powerful collection of graphics utilities for Macintosh, based on Mainstay's acclaimed screen capture utility, graphics and multimedia scrapbook, and graphics viewer. Captivate provides essential graphics utilities to the professional and hobbyist alike.

- Package includes: Captivate Select, Captivate View, and Captivate Store
- Any one of the three can be used alone, and together they make an unbeatable team
- Whether writing a training manual, creating an ad, or just creating a startup screen from your favorite picture, Captivate is everything professionals need at a price anyone can afford.

Our Price **$79** (SCAPTIV)

MULTI MEDIA

CodeWarrior Gold 11

by Metrowerks

- New improved IDE - graphical browser view, new project manager, global preferences and an improved external editor, easy-to-use IDE supports C/C++, Object Pascal and Java
- Full Java toolkit: project manager, linker, editor, class browser, source-level debugger, applet viewer and more! Everything you need for industrial-strength programming
- New Object Pascal cross-compiler and New MPW Object Pascal compiler. Develop for Macintosh, Power Macintosh, Windows 95, Windows NT, Magic Cap and BeOS
- Includes early release ActiveX Support and new VSC interface Version Control Support
- Includes improved JAVA support, an alpha version of the new Version 2.0 CW IDE, on-line books, extensive reference material and Apple Guide files for easy navigation through tutorials and examples
- Two free updates and technical support included with registration

Our Price **$399** (SCWGOLD)

SEE RELATED PRODUCTS: • Code Manager • Inside Power Plant • Inside CodeWarrior 9 • C++ Programming with CodeWarrior • PowerMac Programming Starter Kit • Discover Programming for Macintosh • Learn C on the Macintosh • Metrowerks CodeWarrior Programming

CodeWarrior Wear (XL only)

You live it, you breath it... you might as well wear it!

- Black CodeWarrior Sweatshirt:

Our Price **$29.95** (ACWSWEAT)

- "Blood, Sweat & Code" black short-sleeve shirt:

Our Price **$9.95** (ACWSBLOOD)

- Hawaii Five-0 shirt:

Our Price **$7.95** (ACWHAWAII)

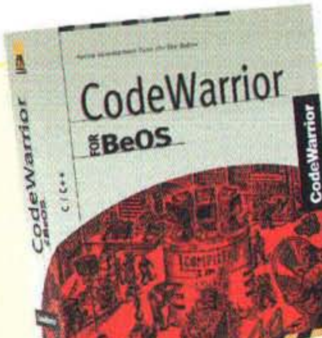
- Hat: Our Price **$14.95**

Please Specify: Black (ACWBHAT) or White (ACWWBHAT)

- Winter Hat (see Web site)

Our Price **$14.95** (AWINHAT)

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.



CodeWarrior for BeOS DR2

by Metrowerks

- Start programming for the new, innovative Be Operating System (BeOS) with complete set of Codewarrior tools
- BeOS-native Integrated Development Environment (IDE) with all the familiar

BeOS™

CodeWarrior features at your fingertips

- A BeOS PowerPC compiler and linker, an editor w/syntax color and styling, and a source-level debugger
- BeOS header and libraries, complete documentation, useful C++ classes, and sample code
- The Be Operating System DR8.2 for Power Macintosh allows you to run and program for the BeOS on a 603 or 604 PCI based PowerMac

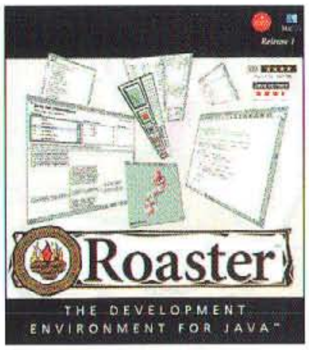
Our Price **$149** (SCWFB)

CodeManager

by Metrowerks

- Source code control system, plug-in to the CodeWarrior IDE
- Compatible with Microsoft Visual SourceSafe version 4.0a
- Cross-platform support (Mac, Windows, UNIX, OS/2)
- Configuration management in excess of 4 billion files
- Over 8,000 files and sub-projects in a single sub-project
- Registered users receive one year of free technical support and two free updates
- **Includes a 1-year MacTech subscription.**

Our Price **$399** (SCDMGR)



Order Toll-free
800-MACDEV-1
(800-622-3381)

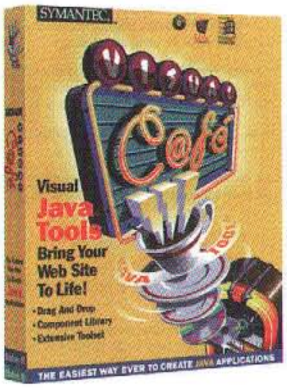
Roaster

by Roaster Technologies, Inc.

Roaster Release 3 is now available and shipping! Get the most out of Sun's Java™ programming language with this powerful development environment.

- Features include: ability to build completely stand-alone Macintosh applications or applets; visual interface builder; ability to create cross-platform zip files; powerful Java debugger; wizard for quickly creating Java applets or applications; JDBC included; Java object database included; ability to call AppleScripts from Java; Just-in-time (JIT) compiler; JDK 1.0.2 support; class tree and hierarchical class browser; much more!
- Software includes: Over 300 example applets and applications; Netscape Internet Foundation Classes; Object Design's PSE for Java; OpenLink Software's JDBC Drivers; OpenSpace Java Generic Library; Microline Component Toolkit Lite 3.0; much more!
- Requirements: Runs on 68k or PowerPC, CD-ROM
- Price includes all web-based updates

List Price $99 **Special Introductory Price! $49** (SROAST3)


SYMANTEC
Visual Café
Visual Java Tools
Bring Your Web Site To Life!
THE EASIEST WAY EVER TO CREATE JAVA APPLICATIONS

Lowest price guaranteed.
Feel Good about your purchase!
30-day return policy.

Symantec Visual Café

by Symantec Corporation

Symantec Visual Café for Macintosh or Windows gives developers the fastest, most productive visual programming environment ever for creating Java applets and applications

- Drag and drop visual programming, easy to learn and use
- Flexible development environment, two-way programming
- Comprehensive component library, create re-usable templates
- Extensive Java toolset and the fastest compilers

Our Price **$199** (SVCAFEMAC)

OR Our Price **$199** (SVCAFEWIN)

SYMANTEC.™

LS Fortran Pro

by Fortner Research LLC

LS Fortran Pro is an ANSI standard Fortran 77 compiler with extensions for Fortran 90, VAX, Cray, Data General, and MS PC Fortran. The package includes both the 68K and Power Macintosh versions of the compiler for MPW, as well as Power Macintosh versions for use in Metrowerk's Codewarrior or Symantec Project Manager.

- Easy to use and produces robust optimized code, wide range of extensions, porting code from mainframe computers is effortless.
- Supports the Macintosh toolbox, provides a full-feature Macintosh output window and offers simple commands for adding a Macintosh user interface.
- Also included: Math77, a library of mathematical subroutines for the computational analysis of scientific data; Source Code Analyzer, a powerful static analyzer that is extremely useful in successfully porting large applications, and for debugging programs with extensive use of Macintosh toolbox calls; and the HDF Libraries, standard data format libraries for effecient storage of many types of ascientific data.

Our Price **$595** (SLSFORT)

LS Fortran Plug-In

by Fortner Research LLC

Codewarrior owners can now experience the power of Fortran. The LS Fortran Plug-In for Codewarrior and Symantec Project Manager, allows programmers to write and compile Fortran programs in the popular, easy-to-use GUI environment, and effortlessly link Fortran with C and Pascal code.

Our Price **$199** (SLSFPI)

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com

F77 SDK
by Absoft Corporation

For Power Macintosh includes a globally optimizing native compiler and linker,native Fx™ multi-language debugger, and Apple's MPW development environment.

- The compiler is a full ANSI/ISO FORTRAN 77 implementation
- Includes all MIL-STD 1753 extensions, Cray/Sun-style POINTER, and several FORTRAN 90 enhancements
- MRWE, Absoft's framework library is included in the MIG graphic library
- Supports the native Macintosh PPC toolbox
- Includes Absoft's Fx debugger which can debug intermixed FORTRAN 77, C, C++, and PPC assembler
- The linker compiler, and debugger all run as native PPC tools, and produce Macintosh PPC executables

Our Price **$589** (SF77)

Pro Fortran
by Absoft Corporation

Absoft Pro Fortran combines native F90, VAX compatible F77, and C/C++ compilers into a single, easy to use environment. All compilers are link compatible and operate through a common interface.

- Graphical debugger, browsers, array display, performance profiler, linker, MRWE application mainframe
- MIG graphics library, Absoft Create Make, several utilities, the latest version of MPW and illustrated documentation
- Whole array operations, modules, interface blocks, and user-defined types or data structures
- Dynamic memory allocation and new control constructs
- F90 is link compatible with Absoft F77, C++, MrC and CodeWarrior
- It is fully compatible with Toolbox, MPW tools, and most third-party products

Our Price **$899** (SAPROF)

NEW PRODUCT!

VIP–BASIC™: Visual Interactive Programming in BASIC
by Mainstay

Now you can create full–featured, stand–alone Macintosh and Power Macintosh applications in just minutes — in standard BASIC code!

- VIP–BASIC™ 2.0 is the fastest way to program your Macintosh.
- Rapid Application Development Environment with Application Framework Included
- Intelligent Dispatcher/Resource Editors and Prewritten Toolbox Calls
- Mix and Match: VIP–BASIC high–level subprograms, standard BASIC libraries
- Macintosh Toolbox calls can be used interchangeably to give you the flexibility
- Includes Powerful Debugger: tightly integrated visual debugger allows program tracing
- Import Pre–Existing BASIC Code: Automatically integrate BASIC code
- Export C Code for Compiling: Automatically convert your BASIC code to C for compilation with Metrowerks CodeWarrior®.
- Includes Full–Featured Mini Database: (limited to 32K) of the powerful VIP–BASIC Database Manager gives you everything you need to setup royalty–free, multi–user database applications.

Our Price **$195** (SVIPBASIC)

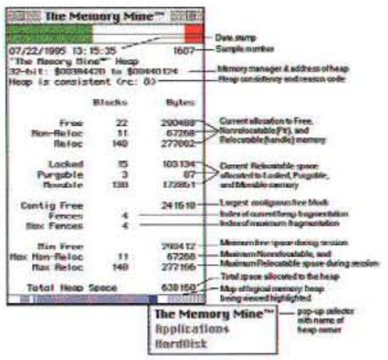
VIP–C™: Visual Interactive Programming in C
by Mainstay

Now you can create full–featured, stand–alone Macintosh and Power Macintosh applications in just minutes. VIP–C™ 2.0 is the first rapid application development system for creating complete Macintosh programs in standard ANSI C. Everything you need is in this box!

- Rapid Application Development Environment: Create full–featured Macintosh apps
- Application Framework Included: set up a bulletproof main event loop and routines
- All source code for the application framework is included with VIP–C
- Intelligent Dispatcher/Resource Editors and Prewritten Toolbox Calls
- Reduce Complexity in Mac Programming with High–level function libraries
- Mix and Match: VIP–C high–level functions, standard C libraries, and Mac Toolbox calls
- Includes powerful, tightly integrated visual debugger
- Import Pre–Existing C Code: Automatically integrate C code with a current project.
- Includes Full–Featured Mini Database: (limited to 32K) of the powerful VIP–BASIC Database Manager gives you everything you need to setup royalty–free, multi–user database applications.

Our Price **$295** (SVIPC)

Memory Mine
by Adianta Inc.



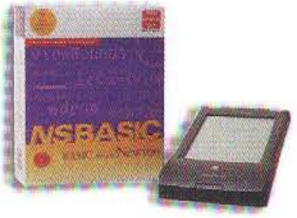
```
The Memory Mine™
07/22/1995 13:15:35              1607
"The Memory Mine™" Heap
32-bit: $00384470 to $00440124
Heap is consistent (rc: 0)

                 Blocks      Bytes

         Free      22       290488
    Non-Reloc      11        67268
        Reloc     140       277002

       Locked      15       103134
    Purgeable       3           87
      Movable     130       172851

  Contig Free               241618
       Fences       4
   Max Fences       4

     Min Free               290412
Max Non-Reloc      11        67268
    Max Reloc     140       277166

Total Heap Space            638150

The Memory Mine™
Applications
HardDisk
```

Date stamp
Sample number
Memory manager & address of heap
Heap consistency and reason code
Current allocation to Free, Nonrelocatable (Ptr), and Relocatable (handle) memory
Current Relocatable space allocated to Locked, Purgeable, and Movable memory
Largest contiguous free block
Index of current heap fragmentation
Index of maximum fragmentation
Minimum free space during session
Maximum Nonrelocatable, and
Maximum Relocatable space during session
Total space allocated to the heap
Map of logical memory: heap being viewed highlighted
pop-up selector with name of heap owner

- Monitor heaps, identify problems such as memory leaks, and stress test applications
- Active status of memory in a heap is sampled on the fly: allocation in non-relocatable (Ptr), relocatable (Handle) and free space is shown, as are heap corruption, fragmentation, and more
- Allocate, Purge, Compact, and Zap memory lets users stress test all or part of a program

List Price $99 Our Price **$94.99** (SMEMMINE)

NS BASIC 3.6 for the Newton with Visual Designer
by NS BASIC Corporation



- A fully interactive implementation of BASIC programming language
- Runs entirely on the Newton – no host is required
- Create files, access the built-in soups, and the serial port f input and output
- Work directly on the Newton, or through a connected Mac/PC and keyboar
- Get the **BASIC Internet Tool**, available at no charge to NS BASIC use from www.nsbasic.com.
- Release Notes with sample code are available from the same location
- Runs on any Newton MessagePad 130 with NS BASIC and the Newto Internet Enabler. Also runs on MP 1201s with NOS 2.0 that have fu memory available
- Write short programs to access News, mail and the web

List Price $99 Our Price **$94.99** (SNSBASIC)

Water's Edge Software

Tools Plus libraries + framework
by Water's Edge Software

Easily create compact, fast running, professional looking applications and plug-ins*. Tools Plus lets you create virtually any user interface element with a single routine, and it transparently provides a robust infrastructure to make all your pieces work together as an application. Rated 4 stars by Macworld!

MacTech (July 96): "it's an incredibly rich collection of tools... If you are interested in developing applications that have 'quality' written all over them, then Tools Plus is for you."

- Simplifies programming and thins source code
- Automates all standard GUI elements
- Thousands of extras, from floating palettes and tool bars to powerful picture buttons
- Includes numerous 3D grayscale options
- Over 1/2 MB of custom fonts, icons, cursors, and other resources
- Includes SuperCDEFs world-class controls (an $89 value) free

Our Price **$249** (STOOLCWG)
CodeWarrior Gold (C/C++ & Pascal, 68K & PPC)
Our Price **$199** (STOOLCWB)
CodeWarrior Bronze (C/C++ & Pascal, 68K)
Our Price **$199** (STOOLSYMT)
Symantec (THINK) C/C++ and THINK Pascal (68K)
Our Price **$149** (STOOLSYM) Symantec (THINK) C/C++ (68K)
Our Price **$149** (STOOLPAS) THINK Pascal (68K)

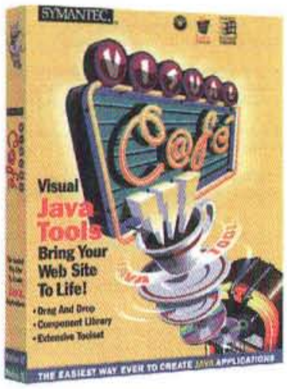
*CodeWarrior required to write plug-ins

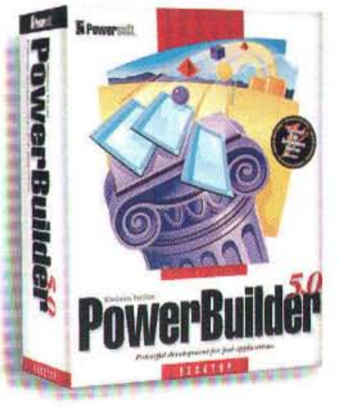
WAIT... There's More!

Here's a list of all available products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	LIST PRICE	OUR PRICE
Discover Programming for Mac	SCWDISCMAC	99.00	79.00
Discover Programming With Java	SCWDISCJAVA	99.00	79.00
LPA MacProlog Developers Edition	SLPAD	1500.00	995.00
LPA MacProlog Programmers Edition	SLPAP	745.00	495.00
Mac FORTRAN II	SFORT2	595.00	549.00
Power MachTen–UNIX	SM10PPC	695.00	695.00
Presenting Magic Cap	BPRESMAGIC	16.95	15.25
Professional Mach Ten for 68k Macs	SPROM10	695.00	695.00
SmalltalkAgents	SSTA	695.00	695.00
Symantec C++ for 68k	SSYMCPP68K	299.00	349.00
Symantec C++ 8.5	SSYMCPP	499.00	349.00
Think Pascal Version 4.0	SPASCAL	199.00	165.00

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com


SYMANTEC
Visual Java Tools
Bring Your Web Site To Life!
THE EASIEST WAY EVER TO CREATE JAVA APPLICATIONS



Symantec Visual Café
by Symantec Corporation

Symantec Visual Café for Macintosh or Windows gives developers the fastest, most productive visual programming environment ever for creating Java applets and applications

- Drag and drop visual programming, easy to learn and use
- Flexible development environment, two-way programming
- Comprehensive component library, create re-usable templates
- Extensive Java toolset and the fastest compilers

Our Price **$199** (SVCAFEMAC)

OR Our Price **$199** (SVCAFEWIN)

PowerBuilder Desktop for Mac
by Powersoft

- Client/Server & Internet Development Tool
- Increases productivity for individual developers
- Point-and-click DataWindow
- Fast complied code, ODBC connectivity to desktop databases
- Anywhere DBMS, create powerful Win, Mac, and Internet apps

Our Price **$295** (SPBDFM)

OOFILE Reporter Writer
by A.D. Software

- Full embedded report-writer, allows you to preview page-by-page and either print or save as plain text, HTML or RTF
- Multiple levels of breaks, database views, headers and footers are provided using a clean object-oriented design
- Incudes RAM-based version of OOFILE database. Included in full OOFILE Platform Bundle
- Saving to file without preview of printing is cross-platform-run on your Mac/Win/Unix server and creates web pages
- Price includes 1-year subscription

Our Price **$500** (SOORW)

ObjectSet Mail SDK for Macintosh
by Smartcode Software

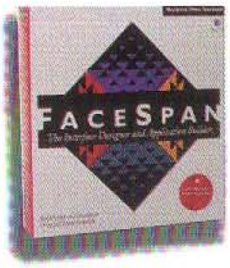
- Provides easy-to-use MIME, SMTP & POP3 APIs
- Royalty-free C++ classes for 68k and Power Macintosh
- Compatible with ALL leading e-mail products
- Ideal for use in Internet and Intranet environments
- Samples with documented, reusable source code
- Windows, UNIX, and multi-platform versions available

Our Price **$495** (SOSMSDK)

BBEdit 4.0
by Bare Bones Software

A powerful, easy-to-learn text editor. Adds new features for HTML coders, including a spelling checker and HTML tag palette. Accelerated for Power Macintosh; dragging supported everywhere; Internet Config aware; PowerTalk aware. Integrated support for Symantec's IDE, Metrowerks CodeWarrior, THINK Reference 2.x, MPW ToolServer, and most other environments. Many UNIX style tools, including "grep" searches, file comparisons, and sorting multi-file search and replace. PopUpFuncs feature lets you jump to a function from a menu.

List Price $119 Our price **$94** (SBBEDIT)

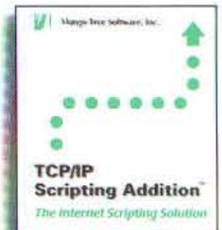


FaceSpan v2.1

by Digital Technology International

- Develop integrated software, make stand alone applications, create friendly interfaces
- Develop quick prototypes, print multiple pages with sophisticated layouts
- Script essential elements of the FaceSpan application - Enhanced save options
- Play and record sounds as either "snd" resources or as "AIFF" files
- Create miniature or complete apps that run on either Power PC or 68k computers
- Use precise time measurement for implementing timed behaviors - New properties
- Proportionally scale PICT images -Align images in a pictbox - Automate any application
- Monitor and respond to low-memory situations-Increased support for Frontier UserTalk!

List Price $299 Our Price **$279** (SFACESPAN)



TCP/IP Scripting Addition

by Mango Tree Software

- Award-winning AppleScript scripting addition
- Allows you to write scripts using MacTCP™ commands in AppleScript™
- Send e-mail or files through a script, check if users are logged on (via Finger), automate FTP, Gopher, NetNews, Telnet, and LPR, verify links in HTML documents, and quickly write many other TCP/IP client-server programs
- Works with AppleScript, MacTCP 2.0.4 and Open Transport

Our Price **$49** (STCP)

WindowScript

by Royal Software, Inc.

WindowScript is the ultimate tool for designing Macintosh user interfaces using HyperCard. Design Real "Macintosh" user-interfaces right inside HyperCard. Until now you either created HyperCard stacks or Macintosh applications. With WindowScript you can literally bring the look and feel of a real Macintosh user-interface to HyperCard. If you're a HyperCard developer, interface designer, application developer, program manager or tester searching for a prototyping tool, WindowScript is perfect for the job.

Our Price **$149** (SWSCRIPT)

Script Debugger

by Late Night Software Ltd.

- A powerful and flexible AppleScript authoring tool – get the most from AppleScript!
- Advanced debugging environment offers single-step script execution with breakpoints
- Script Debugger dictionary browser features a graphical view of objects provided by scriptable applications
- Includes Late Night Software Scripting Additions – a collection of more than 70 new AppleScript commands, and Scheduler, a utility that allows you to launch scripts at pre-determined times

List Price $129 Our Price **$119** (SDEBUG)

Scripter 2.0

by Main Event Software

For professionals, for novices, for webmasters, for solutions providers, there's only one serious choice. Scripter!

- Scripter and FaceSpan work together: one click opens your FaceSpan script in Scripter, another sends it back
- Debug handlers without modifying your scripts using the Call Box
- Applet simulation, live editing, Object map, associated terminology
- Search backwards, block generators, more navigation shortcuts, more drad-and-drop, and an even more enhanced trace log
- ScriptBase is now included; stores your data and media elements (frequently used values, text, pictures, scripts, HTML, headers, file references) and share them between scripts all with a special new browser
- Easily write and compile scripts that have handler declarations and other vocabulary specific to a particular scriptable application
- Scripter is the natural companion to AppleScript for users at all levels of proficiency. Don't write scripts without it!

List Price $199 Our Price **$179** (SSCRIPTER)

AppleScript Finder Guide, English Dialect

by Apple Computer, Inc.

Provides definitions for Finder object classes and commands. Write, record, or run scripts that trigger the same desktop actions that you trigger using the keyboard and mouse.

List Price $19.95 Our Price **$17.95** (BAFG)

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	LIST PRICE	OUR PRICE
DogPatch	SDOGPATCH	299.00	199.00
PreFab Player	SPLAYER	95.00	95.00
ScriptBase	SSCPTBASE	59.00	59.00
ScriptWizard	SWIZ	89.00	84.95
Tenon Ported Applications CD	SPORTED	50.00	49.95

LEGTOP PODEUM™

Legtop Podeum
by Rach Inc.

A combination working platform and carrying case that allows laptop owners a safe and comfortable way to use their computer in a variety of mobile and field environments.

- Straps to your leg for mobility!

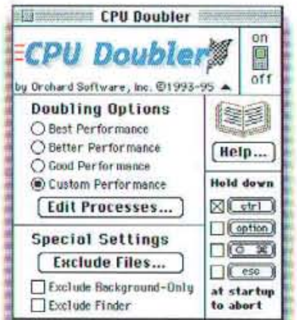
Our Price **$79.95** (ALGTPPOD)

CronManager
by Orchard Software

- Implements the UNIX Cron facility
- Open any Macintosh file on a given date and time
- Simple interface
- Works with any Macintosh file
- Cron Manager bundled with CLImate

Our Price **$26.95** (SCRONMGR)

CPU Doubler
by Orchard Software



CPU Doubler
CPU Doubler
by Orchard Software, Inc. ©1993-95
on / off
Doubling Options
- Best Performance
- Better Performance
- Good Performance
- (●) Custom Performance

Edit Processes...
Help...
Hold down [☒ ctrl] [☐ option] [☐ ⌘] [☐ esc] at startup to abort
Special Settings
Exclude Files...
☐ Exclude Background-Only
☐ Exclude Finder

- Performance enhancement utility for the Macintosh
- Increases the speed of your computer by 100%
- Works on both the PowerPC and 68K Macintosh
- Manages computer throughput using a proprietary scheduling algorithm
- Ensure optimal performance and compatibility

Our Price **$79.95** (SCPU2X)

SEE RELATED PRODUCTS: Development Environments

NeoAccess
by Neologic

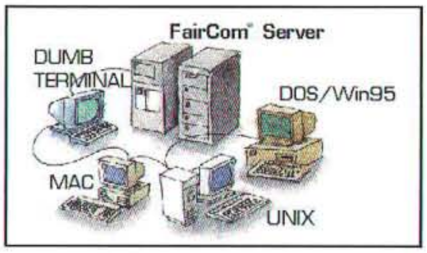
- Full-featured object database engine for use in Macintosh, Windows, Unix and DOS based C++ applications
- Extended binary trees and binary search algorithms tuned for short access times; dynamically combined, collapsed, and compressed indices; object caching for instant access to previously used objects
- Build fast, powerful applications in record time!

Our Price **$749** (SNEO)

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com

c-tree Plus® Database Handler
by Faircom

Unsurpassed Cross Platform Tools for Mac Developers!



- Full C Source
- Client/Server Option
- Over 16 years proven reliability
- Concurrent simultaneous access of Mac/PC files
- Superior throughput and performance
- Unparalleled scalability and flexibility
- Fixed/Variable length files

Our Price **$895** (SCTPDH)

r-tree Report Generator
by Faircom

Handles virtually every aspect of report generation:

- Complete C source
- Complex multi-line reports
- Multi-file access
- Complete layout control
- Conditional page breaks
- Nested Headers and Footers
- Horizontal Repeats

Our Price **$445** (SRTRG)

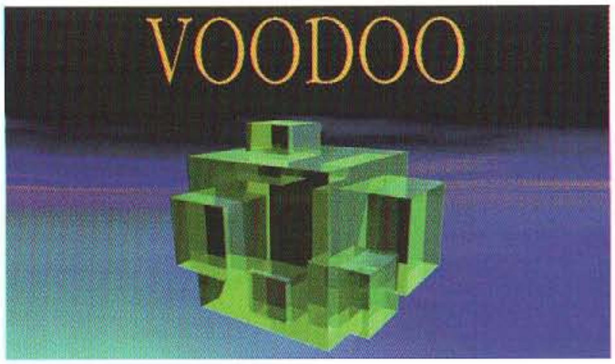
Guide Composer™ 1.2
by StepUp Software

STEPUP SOFTWARE

- Create powerful Apple Guide help systems for any new or existing Macintosh application
- Provides a WYSIWYG development environment: Guide content is developed in Guide windows
- Design topics, phrases, and panels in the same format as the user will use them
- Features are WYSIWYG interface, Topics, phrases, and hierarchical phrases, Coach marks, Fully-Integrated with Apple's Guide Maker (distributed with Guide Composer), compiles scripts automatically, PICTs in Panels, Generated Guide scripts are modifiable
- FREE Update to all registered Guide Composer users. Demo is available at http://www.guideworks.com/

Our Price **$99** (SGCOMP)

SEE RELATED PRODUCTS: AppleGuide Complete, Danny Goodman's AppleGuide Starter Kit, Real World AppleGuide



VOODOO 1.8
by UNI SOFTWARE PLUS

- Smooth integration with Metrowerks CodeWarrior IDE
- Support of AppleEvents and AppleScript
- Comparison of files of different types (not only text files)
- Configurable local file locking (Finder flag or ckid resource)
- Improve handling of local files includeing the creation of folder structures
- Significant performance improvements in many places
- Essential parts PowerPC native
- Version control tool for the simple and clear management of projects in which files
  are created in numerous versions (variants and revisions)
- Allows both variant and revision control, and it manages not only variants and revsions of single files, but of a whole software project (multi files, multi users, multi-variants, access rights, etc.)
- Graphical user interface and is not only suitable for mere source code control but can handle all different kinds of files with amazing compression rates: typical size of delta between arbitrary files 5%
- Please note special prices for multiple copies:
  Single license **$229** (SVOODOO1); 2 pack **$359** (SVOODOO2);
  5 pack **$799** (SVOODOO5); 10 pack **$1369** (SVOODOO10);
  20 pack **$2399** (SVOODOO20)
  Additional pricing available on request.
  SEE RELATED CATEGORY: Dev. Environments

StoneTable 68K/PPC
by StoneTablet Publishing

- StoneTable is a replacement for the Macintosh List Manager
- Available for use with Think C, MPW C & Pascal, CodeWarrior C and Pascal
- Includes libraries for 68K and PowerPC
- An LTable-like class is provided to incorporate StoneTable into the PowerPlant environment
  Our Price **$200** (SSTONEFAT)

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

CompileIt!
by Royal Software, Inc.

CompileIt!, the first HyperTalk compiler, is a complete developement system for the creation of XCMDs and XFCNs.

- Expand the capabilities of your environment by using CompileIt! and the ROM Toolbox extensions
- Increase the speed of routines written in HyperTalk by turning scripts intooexternals
- Protect sensative code from prying eyes because your code is now compiled!
- Easily learn Macintosh programming by exploring the ROM Toolbox
- Includes DebugIt!, a valuable source-level debugger for externals created with CompileIt!
  Our Price **$149** (SCOMPIT)

QC
by Onyx Technology

High performance runtime stress testing for applications.

- Tests include heap checks, purges, scrambles, handle/pointer validation, dispose/release checks, write to zero, de-reference zero as well as other tests like free memory invalidation and block bounds checking
- Extremely user friendly – ideal for non-programmer testers
- Also available in Japanese
  List Price $99 Our price **$94.99** (SQC)

MacA&D 6.0
by Excel Software

MacA&D combines the capabilities of MacAnalyst plus additional detailed design, code generation and code browsing features into one integrated application. It automates structured analysis and design, object-oriented analysis and design, state modeling, task design, data and screen modeling, code editing and browsing, reengineering, requirement traceability and a
multi-user data dictionary. It generates SQL from data models, C++ or Object
Pascal from class diagrams and C, Pascal, Basic or Fortran code from
structure charts.

- Structured analysis and design
- Object-oriented analysis and design
- Real-time and multi-task design
- Data and screen modeling
- Integrated code editing and browsing
- Multi-user dictionary and requirements
- Code to design diagrams for C, C++, etc.
- Design diagrams to code for C, C++, etc.
- State modeling diagrams and tables
- Use cases with traceability
  List Price **$1995** (SMACADP)

SoftPolish CD-ROM
by Bare Bones Software

- The essential tool for software quality assurance on the Macintosh
- Helps you identify inconsistencies with Apple's user interface guidelines, misspelled words, missing resources, and other mistakes
- Provides tools to put the finishing touches on software distribution packages prior to release
- Works independently of any programming language or environment
- Ideal for sanity checking software throughout the development process

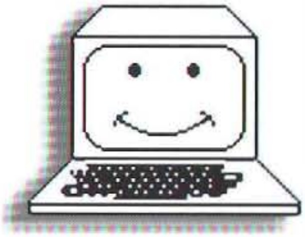
List Price $99 Our price **$89** (SSOFTPOL)

Spellswell Plus 2.0.4
by Working Software

- Award-winning, comprehensive, practical spelling checker that works in batch mode or within applications that incorporate the Apple Events Word Services protocol (e.g., Eudora, WordPerfect, Communicate!, and InfoDepot)
- Checks for spelling errors as well as common typos like capitalization errors, spaces before punctuation, double word errors, abbreviation errors, a/an before vowel/consonant, etc.
- MacTech orders include developer kit with Writeswell Jr., a sample AppleEvents Word Services word-processor and its source code
- Available for OEM Sales

Our Price **$74.95** (SSPELL)

B-Tree HELPER 2.2
by Magreeable Software



- Inexpensive database engine for Macintosh programmers in C source code
- Uses contiguous fixed length blocks
- Expands the file as necessary and contracts files when possible
- Inserts and deletes keys in one or more B-Trees
- Finds keys equal to, less than, or greater than a given value in a few hundredths of a second
- Finds lists of records whose keys are equal to, less than, or greater than a given value or are in a range of values

Our Price **$150** (SBTREE)

Future Basic II
by Staz Software

FutureBASIC II is the award winning leader in Macintosh BASIC programming.

- Source level debugger and Interactive compiler/editor
- Multi-file Project manager and Multi-file find and replace
- Super fast compilation, 32 bit clean, and System 7.x savvy
- QuickBASIC converter
- Getting Started manual with over 500 example files
- Full support of standard BASIC

Our Price **$229** (SFBASIC2)

dtF
by dtF Americas

- True relational database system for Apple Macintosh computers
- Provides a powerful choice for developers who want to create database centered applications with no performance trade-offs
- Features SQL, full transaction control, error recovery, single user, client server architecture and multi-platform support including DOS, Windows, OS/2 and UNIX
- The C/C++ API is identical and fully portable across all supported platforms
- Third-party vendors supporting dtF will be able to offer a variety of advanced features and benefits to their customers royalty free
- Tools are included for importing, exporting, creating and managing databases and users
- Supported development environments include: Symantec, MPW, Metrowerks and more Mac/SDK

List Price $695 Our Price **$679** (SDTF)

Step-Up Installer Pack
by StepUp Software

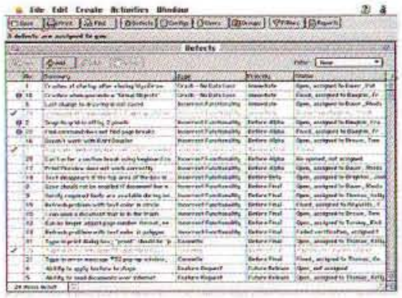
- Package of several Installer "atoms" that let developers incorporate graphics, sounds, file compression and custom folder icons into installation scripts
- Compression formats supported are Compact Pro & Diamond
- Each atom also available separately
- Compression requires additional licensing

Our Price **$219** (SINSTALL)

ScriptGen Pro
by StepUp Software

- Installer script generator which requires no programming or knowledge of Rez
- Supports StepUp's InstallerPack, StuffIt decompression, Compact Pro decompression, custom packages, splash screens, network installs, and resource installation

Our Price **$169** (SSCRPTGEN)



TestTrack-Bug Tracking the Macintosh Way

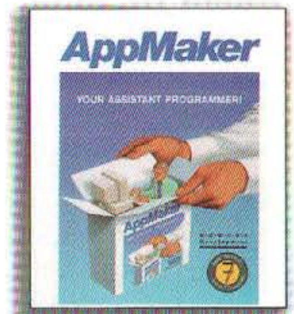
by Seapine Software, Inc.

- Tracks bugs, feature requests, test configurations, users, and more
- Includes notifications, security, a powerful filter mechanism, and multiple reports
- Links your testers, engineers, documentations staff, and project managers together to ensure all bugs are identified, fixed, and documented
- Eliminates the need to build custom bug tracking solutions using general purpose database tools
- Supports single- and multi-user bug databases (additional licenses required to use multi-user features)

Our Price **$129** (STETR)

AppMaker

by Bowers Development



- Develop the user interface for a Macintosh application using the original interface builder
- Just point and click to design your application
- Creates resources and generates excellent source code
- Supports most development environments including Metrowerks, Symantec, or MPW; C, C++, or Pascal; procedural or object-oriented, using PowerPlant, TCL, or MacApp
- The generated code uses the Universal Headers to provide PowerMac compatibility
- Great tool for beginners to learn object-oriented and Macintosh Toolbox programming techniques
- Includes one-year subscription on CD and hardcopy documentation

List Price $299 Our Price **$199** (SAPPMAKE)

MacFlow™: Flowchart Design and Development

by Mainstay

Use MacFlow™ 4.0 for Macintosh to produce top–quality diagrams fast, without tedious drawing. Present your company's organizational structure or design and document the flow of a manufacturing process. Develop any overall view of work, project, or logic flow in just minutes! No other flowcharting method offers the ease–of–use, speed, and outstanding flexibility of MacFlow for Macintosh.

- Create Quality Flowcharts — Fast
- Intelligent Symbols Make Flowcharting — Easy
- Turn Outlines Into Flowcharts — Automatically
- Customize Line and Arrowhead Styles
- Expand Functionality Using Plug–In Modules
- Link Nested Charts, Comments, and Files
- Share Charts Electronically — royalty–free!

List Price **$179** (SMACFLO)

Plan & Track™: Project Planning and Management

by Mainstay

Plan & Track™ 3.5, for Macintosh helps you plan and manage any project. With Plan & Track, you can create a project plan minutes after opening the package. There's no faster or easier way to keep things on track and within budget than with Plan & Track.

- Direct Approach to Project Planning: Quickly plan projects, events, and campaigns
- Create Presentation–Quality Project Plans: Quality, full–color charts or layouts, Integrated Spreadsheet and Graphing
- Automatic Calendar Management: Plan & Track automatically creates and manages, Multiple–Page Schedules
- Earned Value Analysis: Offers a simple method of relating value, cost, and progress
- Drag–and–Drop Support, Standard and User–Defined Symbols and Task Bar Styles

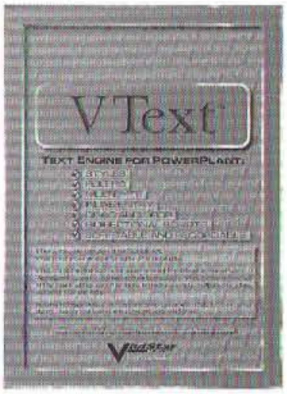
List Price **$179** (SPLNTRK)

Phyla™: Object–Oriented Database

by Mainstay

- Powerful Databases Without Programming: Phyla handles your all your complex database needs
- Define a Database in Minutes: Using an intuitive, graphical user interface
- Objects Are a More Natural Approach: Phyla creates real world databases
- Drag–and–Drop Ease: Relate objects by simply dragging objects between windows
- Create Custom Forms and Reports: Quickly create custom forms and reports
- Fast Finds and Sorts: Perform complex queries and calculations without programming
- Synchronize Multiple Databases Copies
- Password Protection With Access Limitations
- Easy Import and Export: Import from other databases, export data in various formats

List Price **$179** (SPHYLA)



VText

by Vivistar

VText is a C++ add-on library for Metrowerks' PowerPlant application framework. VText provides complete Macintosh text support including: greater than 32kb text, undo, drag and drop editing, AppleEvent scripting and recordability, full support for multibyte characters and inline input methods including Japanese and Chinese text, and full support for bi-directional script systems including Arabic and Hebrew.

- Full featured text engine for Metrowerks' PowerPlant
- Stylesets and rulersets with tabs
- Flexible object oriented C++ API
- Full undo and drag and drop editing
- WorldScript savvy including bidirectional and multibyte scripts with inline editing
- AppleEvent factored for scriptability and recordability

Our Price **$350** (SVTEXT)

LiveAccess™ 1 User Edition

by dtF Americas

Suite of OpenDoc™ parts for accessing relational database systems.

- Compatible with FileMaker Pro, 4th Dimension, Oracle7 and ODBC

List Price **$69** (SLAUE)

LiveAccess™ 1 Developer Edition

by dtF Americas

Suite of OpenDoc™ parts for accessing relational database systems

- Includes the User Edition plus IDL-files to extend the LiveAccess parts suite
- Compatible with FileMaker Pro, 4th Dimension, Oracle7 and ODBC

List Price **$99** (SLADE)

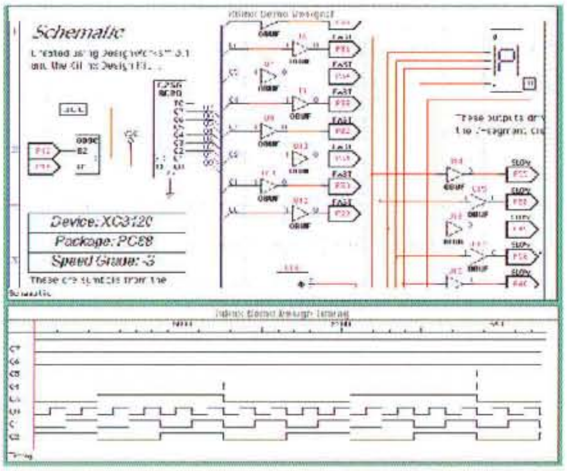
OpenGL for the Macintosh

by Conix Graphics

OpenGL

- Powerful 3D graphics library, 100% OpenGl compliant
- Portable to more platforms than any other API
- Delivers workstation-class performance to the Mac
- RAVE hardware support gives you the speed you need
- Multi-processor capability & works with ALL compilers

Our Price **$279** (SOPENGL)



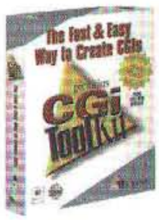
DesignWorks 4.0

by Capilano Computing

DesignWorks 4.0 has all the ease of use and schematic editing power of previous versions, plus new features designed to make your entire design process easier and more error free. The 4.0 version has new menucustomization and scripting features that will directly address your designchecking and interfacing needs.

- Flexible schematic editing features speed the drawing process
- Full Undo/Redo on all editing operations
- Hierarchical design with unlimited levels is fully supported
- Powerful attribute features allow arbitrary text information to be associated with any signal, device or device pin
- Extensive symbol libraries with over 12,000 parts in ANSI and IEEE format
- Integrated device symbol editor allows you to create custom symbols using standard drawing tools
- Interactive digital simulator option is available. No netlists, no application switching!

List Price **$995** (SDWORKS)



CGi Toolkit

by Pictorius Inc.

The Pictorius CGi Toolkit is the fast and easy route to high performance CGIs and ACGIs for your Mac Web site.

- Interactively develop CGIs while the web server, the CGI Toolkit and the browser are running on the same machine
- Interactively develop, test and debug CGIs before compiling
- Powerful debugger allows you to edit code, roll back, code and change input values while your application is running
- Fully object oriented so you can re-use your code
- Automatic handling of Apple Events so you can concentrate on building functionality
- Easy creation of multi-function CGIs which reduces application footprint and RAM usage

List Price **$149** (SCGITLKT)

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com

QUED/M 3.0
by Nisus Software

- The programmer's text editor that defined the industry standard for speed and efficiency
- PowerPC native
- Features integrated support for Symantec C/C++, Metrowerks CodeWarrior 6, and MPW
- Supports all the major development environments on the Macintosh.
- Powerful editing features, including unlimited undo and redo, macro language, scripting, text folding, ten editable/appendable clipboards, markers, displaying text as ASCII codes, dynamic coloring of C/C++ keywords/comments, rectangular and non-contiguous selection
- Includes Celestin Company's APPRENTICE 4

List Price $149 Our Price **$89** (SQUEDM)

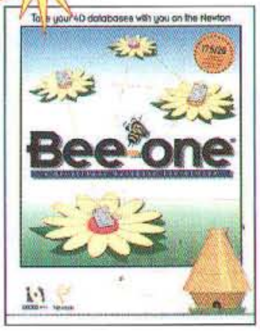
NEW PRODUCT!

Bee-one
by Power Box

Bee-one lightens your load on the road by adapting relational databases developed under 4D® to the Newton Platform. Once the program is installed on both the Macintosh and the Newton, it takes 4 simple steps to use Bee-one!

- Database transfer, Set-up, Use, and Synchronization

Our Price **$139** (SBEEONE)



Order Toll-free
800-MACDEV-1
(800-622-3381)

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	LIST PRICE	OUR PRICE
Animation Class Library	SACL	250.00	250.00
CLImate	SCLIMATE	59.95	59.95
CMaster	SCMASTER	129.95	129.95
DataScript	SWDSCRIPT	249.00	229.99
ICONIX PowerTools-10 Pack	SICPP10	7,995.00	7,845.00
ICONIX PowerTools-6 Pack	SICPP6	5,995.00	5,945.00
ICONIX PowerTools-8 Pack	SICPP8	6,995.00	6,945.00
ICONIX PowerTools-AdaFlow	SICADA	1,495.00	1,395.00
ICONIX PowerTools-ASCII Bridge	SICASCII	1,495.00	1,395.00
ICONIX PowerTools-CoCoPro	SICCOCO	1,495.00	1,395.00
ICONIX PowerTools-DataModeler	SICDATAMOD	1,495.00	1,395.00
ICONIX PowerTools-FastTask	SICFASTTASK	1,495.00	1,395.00
ICONIX PowerTools-FreeFlow	SICFREEFL	1,495.00	1,395.00
ICONIX PowerTools-Object Modeler	SICOBJMOD	1,495.00	1,395.00
ICONIX PowerTools-PowerPDL	SICPOWER	1,495.00	1,395.00
ICONIX PowerTools-QuickChart	SICQUICKCH	1,495.00	1,395.00
ICONIX PowerTools-SmartChart	SICSMART	1,495.00	1,395.00
ICONIX Training & Consulting	TICONIX	3,000.00	2,945.00
IMSL Math and Stat F77	SIMSLSTAT	495.00	495.00
Info-Mac X	SINFOMAC10	39.95	35.95
LJ Profiler	SLJPROF	295.00	295.00
Mac Source II	SMACSOURCE	29.95	26.95
MacA&D 6.0	SMACADP	2,995.00	1,995.00
MachTen Code Builder	SM10CODEB	149.00	149.00
Macintosh Common LISP 4.0	SMCLISP	725.00	725.00
Spellswell Plus 2.0.4	SSPELL	74.95	74.95
Spyer	SSPY	39.00	39.00

Casino!

by BeachWare, Inc.

Can't make it to Vegas this month? Your best bet is Casino! Whether your favorite is Slots, Poker, Blackjack, or Keno, this virtual casino will entertain you for hours with its ten different machines. Mac System requirements: Color Mac, CD-ROM drive, 4 MB of RAM. PC system requirements: Windows 3.1 or later, CD-ROM drive, 4 MB of RAM.

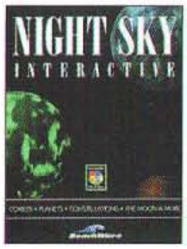
Our Price **$24.95** (SCAS)

Classic Arcade

by BeachWare, Inc.

Ten of your favorite coin-arcade games, redone with killer graphics and sounds! Walk through a virtual arcade and test your game playing skills with these exciting arcade classics. Ten games, including Moon Lander, Astro-Boing, Hyper Hockey, Ballistic Avenger, and more. System Requirements: Mac - Color Mac with 8 MB RAM, CD-ROM drive. PC - 486 with 8 MB RAM, Sound card, SuperVGA,CD-ROM drive.

Our Price **$29.95** (SCLA)

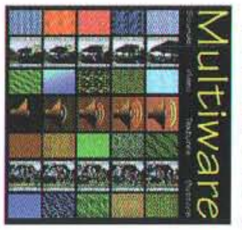


Night Sky Interactive

by BeachWare, Inc.

Introducing a new PC/Mac introductory astronomy learning experience on CD-ROM. Constellations, comets, stars, planets, meteors, galaxies, the moon and more are yours to explore in this CD. Watch animations and hear narratives about how the moon orbits the earth and many other topics. Mac System requirements: Color Mac, CD-ROM drive, 4 MB of RAM. PC system requirements: Windows 3.1 or later, CD-ROM drive, 4 MB RAM.

Our Price **$24.95** (SNSI)



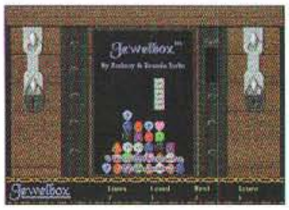
MultiWare Multimedia Collection

by BeachWare, Inc.

Introducing a new Audio multimedia music CD-ROM for the Macintosh. This disc is a collection of clips ideal for Desktop Presentations and other Multimedia applications. This incredible collection of license-free media clips is bursting with 240+ color pictures and backdrops (PICT), 200+ sound & music clips (SoundEdit), 140+ QuickTime movies, and a variety of multimedia tools for use with the Macintosh.

Our Price **$24.95** (SMWMC)

SEE RELATED CATEGORY: Multimedia



1000 Games for Macintosh

by BeachWare, Inc.

The best Macintosh game disc in the entire world, this CD-ROM contains over one thousand great shareware and public domain programs. Battle ugly aliens, blast apart run-away asteroids, deal yourself that royal flush or solve that 3-D puzzle, this disc has it all! System requirements: Mac Plus or greater, CD-ROM drive, and 2 MB of available RAM (4 MB of RAM when running under System 7).

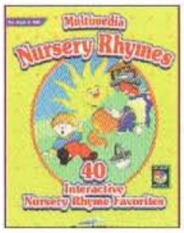
Our Price **$17.95** (STGM)

A Zillion Sounds

by BeachWare, Inc.

An incredible PC and Macintosh sound effects CD-ROM containing 2,177 separate sound effects and short music clips. Each sound is conveniently stored in both SoundEdit and System 7 Beep Sound formats for Macintosh and WAV format for Microsoft Windows. All of the sounds are conveniently organized in separate categories. Mac System requirements: Any Mac, CD-ROM drive, 1 MB of RAM. PC system requirements: Windows 3.1 or later, CD-ROM drive, sound card.

Our Price **$24.95** (SAZS)

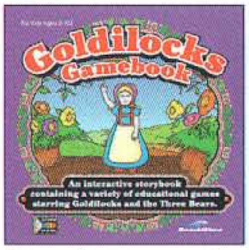


Multimedia Nursery Rhymes

by BeachWare, Inc.

Introducing a magical new PC/Mac playing and learning experience on CD-ROM. This disc contains forty Nursery Rhymes, ten of which include on-screen fingerplays! Each rhyme has its own screen with original artwork, wonderful animation, fun sound effects, and great songs. Mac System requirements: Color Mac, CD-ROM drive, 4 MB of RAM. PC system requirements: Windows 3.1 or later, CD-ROM drive, 4 MB of RAM.

Our Price **$24.95** (SMNR)


Goldilocks
Gamebook
An interactive storybook containing a variety of educational games starring Goldilocks and the Three Bears.

Goldilocks Gamebook

by BeachWare, Inc.

Introducing a beautifully illustrated & animated, interactive version of the classic Goldilocks and the Three Bears story. This PC and Macintosh CD-ROM presents the Goldilocks story in over 30 interactive screens. Throughout the story you can also choose to play one of the seventeen separate games that relate to specific scenes and encounters.

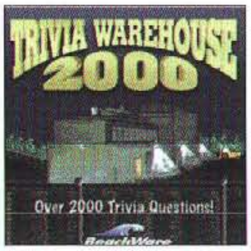
Our Price **$24.95** (SGGB)

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com

Music Tracks
by BeachWare, Inc.

A new PC/Mac & Audio multimedia music CD-ROM. The clips include musical introductions, fanfares, background music, and more. This collection offers you 100 music clips stored in .WAV format for Windows, SoundEdit & AIFF formats for Macintosh and as Audio tracks for audio CS players. All of the music clips are completely license and royalty-free!! Mac System requirements: Mac Plus or greater, CD-ROM drive. PC system requirements: Windows 3.1 or later, Sound Blaster compatible board, CD-ROM drive.

Our Price **$24.95** (SMT)



Trivia Warehouse 2000
by BeachWare, Inc.

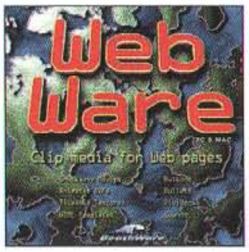
Introducing a fun new PC and Mac CD-ROM. This disc contains two thousand trivia questions, in 45 categories, in several game formats! Test your memory with the Q&A, Concentration, and Multiple Choice games! Categories include: Animals, Bodies, Bond, Bugs, Cities, Comics, Geography, Gilligan, Gross, Health, Holidays, Horrors, Kids, Knot's Landing, Math, Movies, and many more. Mac System requirements: Color Mac, CD-ROM drive, 4 MB of RAM. PC system requirements: Windows 3.1 or later, CD-ROM drive, 4 MB of RAM.

Our Price **$24.95** (STW2K)

Screen Machine
by BeachWare, Inc.

Personalize your computer screen with this dynamic and useful collection of Screensavers and Wallpapers. Choose from over 100 original Screensavers such as flying airplanes, bouncing coffee cups, falling climbing gear, shifting psychedelic patterns, or floating spaceships. Customize your computer desktop with over 150 exciting new wallpapers such as sand, rocks, cloth, coins, cartoons, or unique patterns. Included is an easy to use browser program that lets you sample all of the Screensavers and Wallpapers before installing them on your computer.

Our Price **$24.95** (SSM)



Web Ware
by BeachWare, Inc.

The ultimate collection of clip media and templates for building your own Web Page. An incredible selection of Shockwave movies, animated GIFs, buttons, bullets, dividers, and sample HTML pages. There are literally thousands of graphical elements on this disc, all there to spice up your web page. In all, it's about 300 megabytes of creativity only a mouse-click away! System Requirements: PC - 486 or better with 8 MB RAM , Sound card, SuperVGA, CD-ROM drive. Macintosh - Color Mac with 8 MB RAM, CD-ROM drive.

Our Price **$24.95** (SWEBW)

WAIT... There's More!

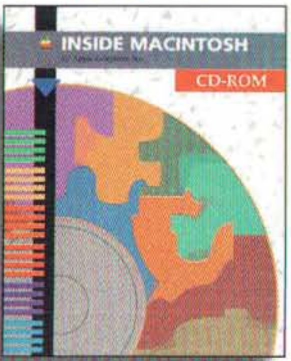
Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	OUR PRICE
3D Game Machine	S3DGAME	$299.00
Digital Swimsuit Portfolio	SDSP	24.95
Do It Youself Credit Repair	SCR	24.95
Nature Photos	SNP	24.95
Photo Textures	SPT	24.95
Religion Bookshelf	SRB	24.95
Vitamin	SVITAMIN	24.95
World War II Almanac	SWW2A	24.95

INSIDE MACINTOSH

by Apple Computer, Inc.

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.



Inside Macintosh: CD-ROM

by Apple Computer, Inc.

More than 25 volumes in electronic form. Includes: QuickDraw™ GX Library, Macintosh Human Interface Guidelines, PowerPC System Software, Macintosh Toolbox Essentials and More Macintosh Toolbox, QuickTime and QuickTime Components. Access over 16,000 pages of information with Hypertext linking and extensive cross referencing.

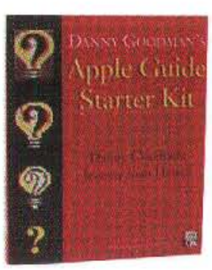
List Price $99.95 Our Price **$89.95** (BIMCD)

WAIT... There's More!

Here are all of the Inside Macintosh products – 10 % off! For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us.

PRODUCT	CODE	LIST PRICE	OUR PRICE
Inside Macintosh: AOCE Applications Interface	BIMAOCE	$44.95	$40.45
Inside Macintosh: AOCE Service Module	BIMAOCES	29.95	26.95
Inside Macintosh: Devices	BIMDEV	29.95	26.95
Inside Macintosh: Files	BIMFIL	29.95	26.95
Inside Macintosh: Imaging with QuickDraw	BIMIMAG	32.95	29.65
Inside Macintosh: Interapplication Communications	BIMIAPP	36.95	33.25
Inside Macintosh: Macintosh Toolbox Essentials	BIMTBOX	39.95	35.95
Inside Macintosh: Memory	BIMMEM	24.95	22.45
Inside Macintosh: More Macintosh Toolbox	BIMMAC	34.95	31.45
Inside Macintosh: Networking	BIMNET	29.95	26.95
Inside Macintosh: Operating System Utilities	BIMOPSU	28.95	26.05
Inside Macintosh: Overview	BIMOVER	24.95	22.45
Inside Macintosh: PowerPC Numerics	BIMPPCNUM	28.95	26.05
Inside Macintosh: PowerPC System Software	BIMPPCSYS	24.95	22.45
Inside Macintosh: Processes	BIMPROC	22.95	20.65
Inside Macintosh: QuickDraw GX Environ. & Utilities	BIMGXENV	31.95	28.75
Inside Macintosh: QuickDraw GX Graphics	BIMGXGR	31.95	28.75
Inside Macintosh: QuickDraw GX Objects	BIMGXOBJ	31.95	28.75
Inside Macintosh: QuickDraw GX Printing	BIMGXPRNT	29.95	26.95
Inside Macintosh: QuickDraw GX Printing Extensions	BIMGXEXT	29.95	26.95
Inside Macintosh: QuickDraw GX Prog. Overview	BIMGXOV	24.95	22.45
Inside Macintosh: QuickDraw GX Typography	BIMGXTYP	29.95	26.95
Inside Macintosh: QuickTime	BIMQT	29.95	26.95
Inside Macintosh: QuickTime Components	BIMQTCOM	34.95	31.45
Inside Macintosh: Sound	BIMSOUND	29.95	26.95
Inside Macintosh: Text	BIMTEXT	39.95	35.95
Inside Macintosh: X-Reference	BIMXREF	19.95	17.95

(Book sale prices are contingent upon availability)

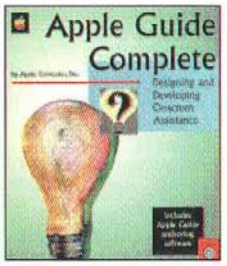


Danny Goodman's Apple Guide Starter Kit

by Danny Goodman and Jeremy Joan Hewes

Create your own Apple Guide databases quickly and easily, without having to learn a scripting language, write coded files, or use several different files and programs. Includes advice and tips on how to design a good Guide, from planning and creation through testing, revising, and indexing. Book/disk, 320 pages.

List Price $34.95 Our Price **$31.46** (BDGAGSK)



AppleGuide Complete

by Apple Computer, Inc.

Includes CD-ROM!

Covers Guide Maker, the software you use to build and test guide files. Learn about the complete cycle of designing as well as advanced topics such as scripting and coding guide files. Book/CD-ROM, 544 pages.

List Price $39.95 Our Price **$35.96** (BAPLGD)

SEE RELATED CATEGORY: Tools, Libs & Utilities

Essential OpenDoc

by Jesse Feiler and Anthony Meadow

Gives an in-depth look at the technical issues of OpenDoc. Explores the three core technologies that support it's functionality – SOM, OpenDoc's storage mechanism, and the Open Scripting Architecture (OSA). Also examines CyberDog, a set of OpenDoc part editors that provides access to Internet services and offers compelling example of the power of OpenDoc development.

List Price $39.95 Our price **$35.95** (BESOD)

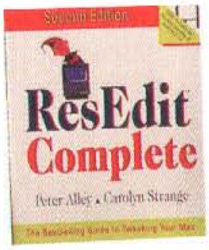
Mac OS 8 Revealed

by Tony Francis

Includes CD-ROM!

The first authoritative look at this exciting new operating system. A must for Mac developers who want to make their software compatible with Mac OS 8. Essential for system administrators who plan to upgrade their system. Included CD-ROM contains demos of new Mac OS 8 features.

List Price $34.95 Our Price: **$31.45** (BMACOS8R)



ResEdit Complete, Second Edition

by Peter Alley and Carolyn Strange

Customize every aspect of your interface form creating screen backgrounds and icons to customizing menus and dialog boxes. Book/disk package, 608 pages.

List Price $34.95 Our Price **$31.45** (BRESED2)

OpenDoc Programmer's Guide

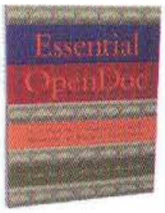
by Apple Computer, Inc.

The official reference for the implementation of OpenDoc on the Mac OS. Describes the component software revolution and explains how to develop for it on the Mac OS platform. Accompanying CD-ROM contains a complete reference to the OpenDoc programming interface, and an extensive collection of tested, reusable sample code.

List Price $44.95 Our Price **$40.46** (BOPENDOC)

OpenDoc Programmer's Cookbook

by Apple Computer, Inc.



Shows you how to create OpenDoc software components, called parts editors, for the Mac OS Platform. Including instructions for setting up the Macintosh Programmers Workshop (MPW) development environment to write OpenDoc software. Annotated listings of explaining the methods that implement the SamplePart part editor. Descriptions of other sample part editors created by the OpenDoc engineering team to illustrate more advanced features. An Introduction to the System Object Model (SOM) technology underlying OpenDoc.

List Price $24.95 Our price **$22.45** (BODCOOK)

The ResEdit All Night Diner

Includes Disk!

by David Ciskowski

An idea-filled menu and introduction to the joys of customizing software. Add personality to the Mac by customizing default icons, the text of menus and dialog boxes, cursors, pointers and more. Disk features ResEdit, plus lots of sample resources.

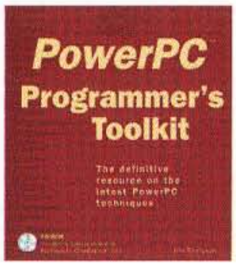
List Price $24.95 Our Price **$22.45** (BRESDINE)

SEE RELATED CATEGORY: Dev. Environments

WAIT... There's More!

Here are some popular books for the NEXTSTEP enthusiast - 10% off!

PRODUCT	CODE	LIST PRICE	OUR PRICE
NEXTSTEP Development Tools & Techniques	BNEXTDTT	$30.95	$27.85
NEXTSTEP General Reference	BNEXTGR	44.95	40.45
NEXTSTEP Programming Interface Summary	BNEXTPIS	30.95	27.85
NEXTSTEP User Interface Guidelines	BNEXTUIG	24.95	22.45



PowerPC Programmer's Toolkit
by Tom Thompson

CD-ROM includes a special version of Metrowerks CodeWarrior 7.0 and sample code from the book. Details how to write PowerPC applications in native code for blazing speed. Written by an Apple Insider.

List Price $45 Our Price **$40.50** (SPPCPT)

Network Frontiers Bundle
by AP PROFESSIONAL

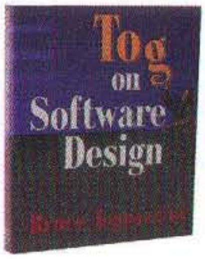
Best Deals On Books! All Prices Reduced 10%

Includes 3 books:

Complete Guide to MAC,Backup Management, Designing AppleTalk Network Architectures, and Managing AppleShare and Workgroup Servers Apple Certified. Each bundle includes the first 3 books that correspond to the Apple Certified Server Engineer (ACSE) program.

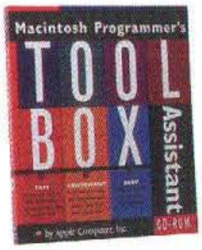
List Price $89.95 Our Price **$59.95** (BNETFB)

Tog on Software Design
by Bruce "Tog" Tognazzini



Respected industry futurist, Tog, presents his vision of our technological future, detailing the steps computer professionals need to take to deliver new technologies that will profit the industry and benefit society in general. Contains Tog's insights on a wide range of topics from quality management to the meaning of standards.

List Price $29.95 Our Price **$26.95** (BTOG)



Programmer's Toolbox Assistant CD-ROM

Includes CD-ROM!

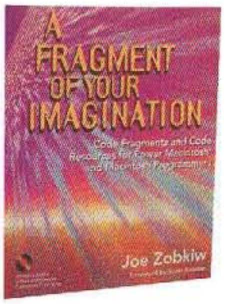
Instant electronic access to Inside Macintosh essentials.
by Addison-Wesley Publishing

Get quick access to reference pages for over 4,000 Toolbox calls in your system software from their development environment. Essential information for Macintosh software developers. Hypertext links allow programmers to view related topics easily. The ultimate electronic reference tool for Macintosh programmers.

List Price $99.95 Our Price **$89.95** (STBASST)

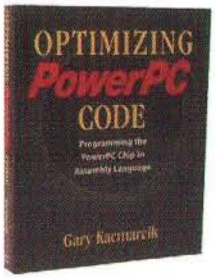
A Fragment of Your Imagination
by Joe Zobkiw

Includes CD-ROM!



Packed with useful code fragments for the Macintosh and Power Macintosh. Hard to find information about techniques used to structure and build fat, safe fat, and accelerated code resources. All code is reusable and is provided on the disc, along with Metrowerks Code Warrior Lite. Book/CD-ROM, 528 pages.

List Price $39.95 Our Price **$35.96** (BFRAG)



Optimizing PowerPC Code: Programming the PowerPC in Assembly Language
by Gary Kacmarcik

Take full advantage of the potential of the PowerPC by mastering the Assembly Language techniques. Learn to produce faster more robust software!

List Price $39.95 Our Price **$35.96** (BOPTPPC)

Macintosh Programming Secrets 2nd edition
by Scott Knaster and Keith Rollin

Macintosh Programming Secrets is divided in two parts

Part 1: "Concepts and Ideas." Discusses the evolution of the Macintosh and the standards, customs, and software that shape the system as well as the Macintosh user interface.

Part 2: "Technical Adventures." Presents the skeleton of an application, and then builds upon that framework to describe how to: Create fancy dialogue boxes. Utilize the new 32 bit QuickDraw developments. Track the mouse with "marching ants".

List Price $31.95 Our Price **$28.76** (BPSECRET)

BASIC for the Newton
by John Schettino & Liz O'Hara

Includes Disk!

Program on Macintosh, Windows-based PC, or on the Newton itself. Straight-forward "programming by example" approach – you'll be writing Newton programs right away. Includes 3.5" disk containing Demonstration NS BASIC and over fifty example programs (Newton not included).

List Price $35.95 Our Price **$32.35** (BNEWT)

SEE RELATED CATEGORY: Dev. Environments

Newton

Programming for the Newton: Software Development using NewtonScript
by Julie McKeehan and Neil Rhodes.
Foreword by Walter R. Smith

An indispensable tool for Newton programmers. Includes disk with sample Newton application from the books, as well as demonstration version of Newton Toolkit (NTK) – the complete development environment for the Newton®. A Publication of AP Professional May 1994, Softcover, 393 pages.

List Price $29.95 Our Price **$26.96** (BPROGNEWT)

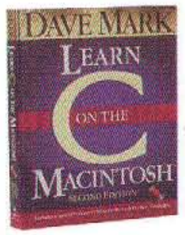
SEE RELATED CATEGORY: Dev. Environments

Programming with AppleTalk
by Michael Pierce

Programming with AppleTalk is the hands-on guide to understanding and working with AppleTalk - Topics include: How to create applications and system extensions that run with AppleTalk. AppleTalk protocols and the protocol stack, transport media, Preferred AppleTalk Inteface, and the storage management. Numerous working code examples walk you through using RDEV, INIT, NBP, ATP, and ADSP

List Price $24.95 Our Price **$22.45** (BPROAT)

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com



Learn C on The Macintosh Second Edition
By Dave Mark

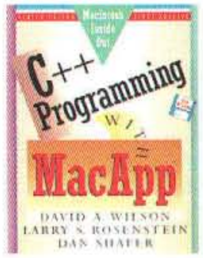
New revised edition! Easy-to-understand – everything you need to start programming. Updated and enhanced exercises that lead you step by step. You'll learn function, variables, point datatypes, data structures, file input and output and more! Includes CD-ROM with Metrowerks CodeWarrior™ Lite.

List Price $34.95 Our Price **$31.45**
(BLEARNC2)
SEE RELATED CATEGORY: Dev. Environments

Learn C++ on the Macintosh
by Dave Mark

Basic syntax of C++ and object programming. Learn how to write, edit, and compile your first C++ programs. Features key C++ concepts such as derived classes, operator overloading, iostream functions and more. Includes a special version of Symantec C++ for Macintosh. Book/disk package with 3.5" 800K Macintosh disk. 400 pages.

List Price $38.95 Our Price **$35.05**
(BLRNCPP)
SEE RELATED CATEGORY: Dev. Environments



C++ Programming with MacApp
by David Wilson, Larry Rosenstein & Dan Shafer

Learn the secrets to unlocking the power of MacApp®, Apple's development environment for C++. Learn to design complex windows and views using the ViewEdit tool. Learn to support multipage text and graphics with only five lines of code.

List Price $34.95 Our Price **$31.46**
(BCPPMACAP)
SEE RELATED CATEGORY: Tools, Libs & Utilities

Object Oriented Program Design
by Mark Mullin

A concise guide to the concepts and techniques of OOP design. Clarifies the key concepts of object oriented programming such as objects, classes, entities, hierarchies, and inheritance. Uses typical database application to illustrate each OOP topic, to give a familiar point of reference.

List Price $22.95 Our Price **$20.66**
(BOOPRODES)

Perl Quick Reference
by Michael O. Foghlu

Commands are sorted by task, class packet, platform or hardware compatability all in alphabetical order. Includes jump tables to guide reader to specific pages in the reference. Designed in a larger trim size than traditional Quick References-and with a lay-flat binding.

List Price $19.99 Our Price **$17.99**
(BPERLREF)

OBJECTIVE-C Object-Oriented Programming Techniques
by Lewis J. Pinson and Richard S. Wiener

Presents the basic concepts of object-oriented design and programming, and provides a precise description of the Objective-C language. Several small- to medium-sized applications using Objective-C illustrate the general principles of object-oriented programming. Covers the two main versions of the Objective-C language. Demonstrates the versatility and power of the NeXT machine as a platform to support object- oriented programming. **Shows how to design, implement, and use hierarchies of classes. Explains the purpose of pre-defined classes and shows how they can be used in designing programs.**

List Price $37.95 Our Price **$34.15**
(BOBJCOOPT)

Macintosh C Programming Primer Volume I
Second Edition, Inside the Toolbox Using THINK C
by Dave Mark and Cartwright Reed

Updated new edition of the Macintosh programming best seller. System 7, new versions of THINK C and ResEdit. Learn how to use the resources, Macintosh Toolbox and interface to create stand-alone applications. 672 pages.

List Price $26.95 Our Price **$24.25** (BCPRIM1)

Macintosh C Programming Primer Volume II
Mastering the Toolbox Using THINK C
by Dave Mark

Covers advanced topics such as: Color QuickDraw, THINK Class Library, TextEdit, and the Memory Manager. 528 pages.

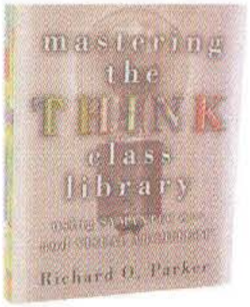
List Price $26.95 Our Price **$24.25** (BCPRIM2)

MacsBug Reference & Debugging Guide
For MacsBug version 6.2
by Apple Computer, Inc.

Includes CD-ROM!

MacsBug is an assembly-language-level debugging tool. Includes macros, templates, dcmds, and other resources for making debugging easier. Covers how to display and set memory and process registers, disassemble memory, and set execution breakpoints. Explains Discipline, a tool for testing the validity of toolbox parameters, and debugging strategies to find & cure common bugs. Includes MacsBug 6.2.

List Price $34.95 Our Price **$31.46**
(BBUGREF)



Mastering the THINK Class Library
by Richard Parker

Provides a thorough examination of Symantec's extensive Class Library and the Visual Architect. A complete description of the structure and operation of the TCL includes explanations of all code generated by the Visual Architect, any necessary custom code, and the operation of this code. Visual Architect tutorials provide you with a step-by-step approach for simplifying the development of complex Macintosh applications, 496 pages.

List Price $29.95 Our Price **$26.95** (BMASTERTCL)

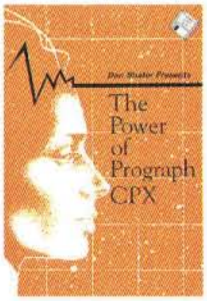
SEE RELATED CATEGORY: Dev. Environments

Visual Programming with Prograph CPX
by Scott B. Steinman and Kevin G. Carver

An introduction to the language and a guide for advanced users, for both Macintosh and Windows-based machines.

List Price $34 Our Price **$30.60** (BVISPRO)

SEE RELATED CATEGORY: Dev. Environments



The Power of Prograph CPX
by Dan Shafer

Master the revolutionary graphical object-oriented programming language. Step-by-step course through three interrelated projects of increasing complexity. Learn Prograph language, CPX classes and object editors. Includes disk with all code in the book.

List Price $49.95 Our Price **$19.95** (BDANPRO)

Inside CodeWarrior 11
by Metrowerks

Includes CodeWarrior IDE User's Guide. This is the printed version of the documentation provided on the CD. Covers CodeWarrior, the debugger and associated tools.

Our Price **$34.95** (BINSCW)

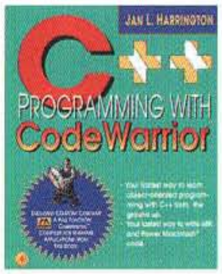
SEE RELATED CATEGORY: Dev. Environment

Metrowerks CodeWarrior Programming
by Dan Parks Sydow

Includes CodeWarrior Lite, and Full Coverage of PowerPlant™. The best information on Metrowerks CodeWarrior, giving full coverage to the Gold Edition. CD includes Code Warrior Lite.

List Price $39.95 Our Price **$35.95** (BCWPROG)

Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com



C++ Programming with CodeWarrior
by Jan L. Harrington

Beginning OOP for the Macintosh and Power Macintosh and Mac OS compatibles. Learn object-oriented programming techniques using C++ as the example language and Metrowerks and CodeWarrior as the example compiler. Enclosed CD contains example code from the book and a full-function Metrowerks CodeWarrior.

List Price $35.95 Our Price **$32.35** (BCPPCW)

CodeWarrior Software Development Using PowerPlant
by Jan L. Harrington

C++ programmers will learn to develop object-oriented software applications for the Mac and Power Mac using the PowerPlant environment and the classes that support it. Covers CodeWarrior 8. Included CD-ROM contains source code for all the programming examples in the book and Metrowerks CodeWarrior Lite.

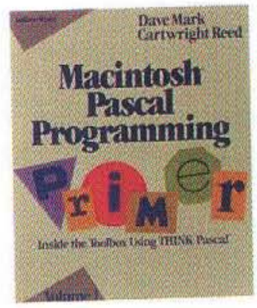
List Price $34.95 Our Price **$31.45** (BCWSWDEV)

Inside PowerPlant Manual
by Metrowerks

Create PowerPlant applications using the CodeWarrior IDE and PowerPlant Constructor. Full descriptions of major PowerPlant classes and resources. Included are the PowerPlant Constructor Manual, including View, TextTraits and Custom Types editing, and PowerPlant Library Reference, covering all classes and functions in PowerPlant.

Our Price **$34.95** (BINSPP)

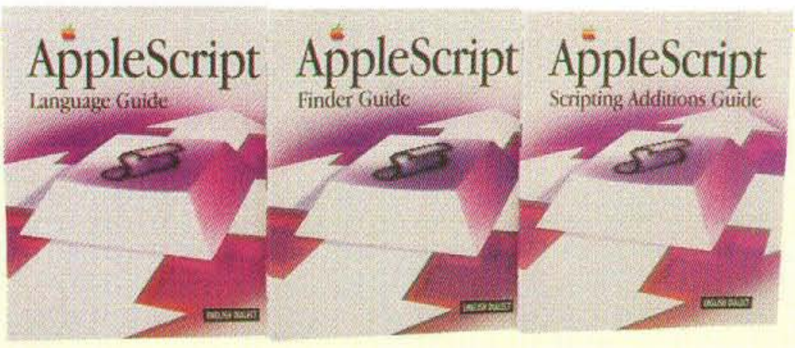
SEE RELATED CATEGORY: Dev. Environment



Macintosh Pascal Programming Primer Volume I
Inside the Toolbox Using THINK Pascal
by Dave Mark and Cartwright Reed

This tutorial shows programmers new to the Macintosh how to use the Toolbox resources, and the Macintosh interface to create stand-alone applications with Symantec's THINK Pascal. 544 pages.

List Price $26.95 Our Price **$24.25** (BPASCPRI)



AppleScript Language Guide
by Apple Computer, Inc.

A complete reference for anyone using AppleScript to modify existing scripts or to write new ones. Contains useful information for programmers who are working on scriptable applications or complex scripts. Features detailed definitions of AppleScript terminology and syntax in the following categories: Value classes, commands, objects and references to objects, expressions, control statements, handlers, and script objects. Includes many sample scripts, discusses advanced topics such as writing command handlers for script applications, the scope of script variables and properties declared at different levels in a script, and inheritance and delegation among script objects.

List Price $29.95 Our Price **$26.95** (BALG)

SEE RELATED CATEGORY: Scripting

Order Toll-free
800-MACDEV-1
(800-622-3381)

AppleScript Applications: Building Applications with FaceSpan and AppleScript
by John Schettino Affiliation & Liz O'Hara

Build complete AppleScript applications using FaceSpan, a user interface development tool that makes AppleScript applications truly "Mac-Like". Uses a step-by-step approach to demonstrate techniques for building applications through illustrations and samples. Provides Graphical User Interface (GUI) design tips and practical approaches for implementation. Contains one CD-Rom with AppleScript 1.1, a demonstrations version of FaceSpan 2.1, source code for all example applications numerous AppleScript shareware and demonstrations programs. Contains a section on debugging AppleScript applications using FaceSpan.

List Price $34.95 Our Price **$31** (BAPSCAP)

Applied Mac Scripting
by Tom Trinko

Learn to design and develop powerful scripts. Covers AppleScript™, Frontier, QuicKeys, Tempo II, nShell, FaceSpan Application Builder, Scripting PlainTalk and System 7.5. Hands on tutorial shows you how to automate your Macintosh activities by learning how to use the AppleScript and Frontier scripting environments. Harness the capabilities of a wide variety of Macintosh applications into the integrated productivity tools. This includes such things as the newspaper script which combines the power of SITcomm, MacWrite Pro, and FileMaker Pro, or QuarkXPress.

List Price $34.95 Our Price **$31.45** (BAPPLIED)

SEE RELATED CATEGORY: Scripting

AppleScript Scripting Additions Guide
by Apple Computer, Inc.

Use the standard scripting additions commands. Write scripting additions.

List Price $18.95 Our Price **$17.05** (BSCRADD)

SEE RELATED CATEGORY: Scripting

Lowest price guaranteed. Feel Good about your purchase! 30-day return policy.

The Elements of E-mail Style
Communicate Effectively via Electronic Mail
David Angell
Brent Heslop

The Elements of E-Mail Style
by Brent Heslop and David Angell

Write solid, effective E-Mail and avoid common pitfalls.

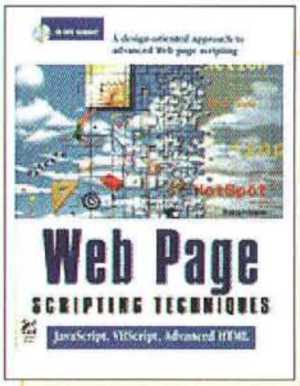
List Price $14.95 Our Price **$13.45** (BEMAIL)

AppleScript Finder Guide, English Dialect
by Apple Computer, Inc.

Provides definitions for Finder object classes and commands. Write, record, or run scripts that trigger the same desktop actions that you trigger using the keyboard and mouse.

List Price $19.95 Our Price **$17.95** (BAFG)

SEE RELATED CATEGORY: Scripting

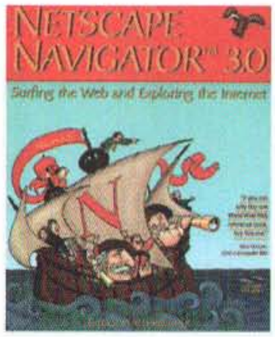


Web Page Scripting Techniques
by MacMillan

Manipulating color effectively. Placing images and using shapes. Using JavaScript functions and objects, plus VBScript functions and objects. Determining users browser type and operating system.

List Price $50 Our Price **$45** (BWEBPST)

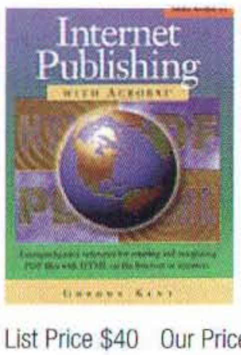
Check out our Web site!
• Full product descriptions • Hundreds of more products
http://www.devdepot.com



Netscape Navigator 3.0
by Bryan Pfaffenberger

Learn the best Web surfing techniques. Quickly find your way to valuable information on the Web. Learn to use the new plug-in tools, including Live3D and Real Audio. Send and receive electronic mail to anyone on the Internet. Utilize the enhanced security features for on-line shopping and ordering.

List Price $29.95 Our Price **$26.96** (BNETN3)



Internet Publishing with Adobe Acrobat
by MacMillan

Optimizing PDF for Web viewing, integrating HTML and PDF, selecting PDF page dimensions, PDF file evaluation. Setting links to PDFs and URLs, PDF forms, Security algorithms, Configuring Web servers for PDF, Embedded PDFs, PDF creation techniques.

List Price $40 Our Price **$36** (BIPWAA)

Netscape Navigator Starter Kit for Macintosh
by MacMillan

Create your own home page in minutes and add hyperlinks to your favorite web sites. Add graphics, audio, and video. Download files via FTP. Load plug-ins to enable movies, audio, animations, and more.

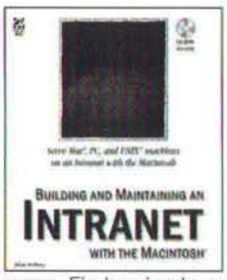
List Price $34.99 Our Price **$31.49** (BNETNSK)

Providing Internet Services via the MacOS
by Carl Steadman and Jason Snell

Includes CD-ROM!

Shows you how to provide Web pages, FTP, Gopher, e-mail, and more with the Mac OS. Includes CD-ROM containing all the Mac server software and utilities you need.

List Price $34.95 Our Price **$31.46** (BPROVNET)

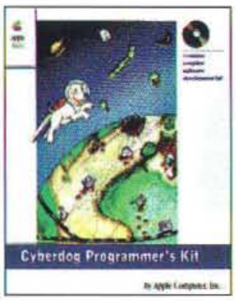


Building and Maintaining an Intranet with the Macintosh
by Tobin Anthony

Includes CD-ROM!

Set up services such as video conferencing, Real Audio, message boards, employee scheduling, e-mail, WWW, ftp, and many more. Find a simple, cost-effective solution to providing corporate information to company employees. Learn how to open portions of your intranet site up to the internet while keeping private information safe. Companion CD-ROM.

List Price $50 Our Price **$45** (BBAMAI)

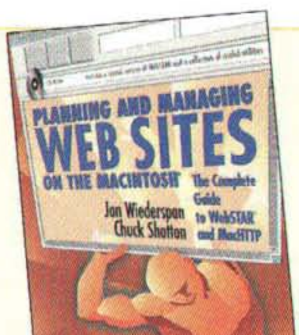


Cyberdog Programmers Kit
by Apple Computer, Inc.

Includes CD-ROM!

Apple's official reference. A must for developers who want to make customizable Internet access available to all Mac users. Included CD-ROM contains all the tools needed to create Cyberdog-aware components.

List Price $39.95 Our Price **$34.95** (BCYBERDOG)

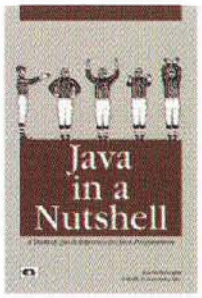


Planning and Managing Websites
by Jon Weiderspan and Chuck Shotton

Includes CD-ROM!

The definitive guide to setting up and running a Web site on the Macintosh. Learn everything you need to know about using WebSTAR, the best known HTTP server software and its shareware predecessor MacHTTP. Write CGI applications for your server – in AppleScript and in C. CD includes a special version of WebSTAR, plus tons of useful software

List Price $39.95 Our Price **35.96** (BPLANWEB)



Java in a Nutshell
by David Flanagan

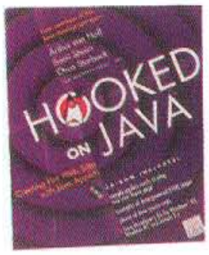
A complete quick reference guide to Java, the hot new programming language from SunMicrosystems. Contains descriptions of all of the classes in the Java 1.0 API, with a definitive listing of all methods and variables. Also contains an accelerated introduction to Java for C and C++ programmers who want to learn the language fast.

List Price $14.95 Our Price **$13.45** (BJAVANUT)

JavaScript for the Macintosh
by Matt Shobe and Tim Ritchey

Allows non-programmers to take advantage of the power of Netscape Navigator. Expand the capabilities of your Web page, without having to understand C or C++. CD-ROM contains "Wizlets" that allows you to easily create your own JavaScripts. Takes you step-by-step through programming cross-platform JavaScripts. Details how to create JavaScripts for JavaScript-aware Web browsers.

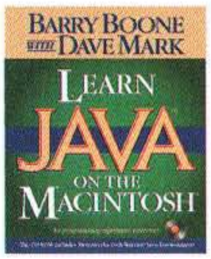
List Price $45 Our Price **$40.50** (BJAVASCRPTJ)



Hooked on Java
by Arthur Van Hoff, Sami Shaio and Orca Starbuck

Written by the Java development team at Sun Microsystems, Inc. An introduction to using applets, for Web administrators, designers, and developers. Demonstrates how to use applets in your own pages. Includes a concise introduction to the Java language, and a CD with tools.

List Price $29.95 Our Price **$26.95** (BHJAVA)



Learn Java on the Mac
by Barry Boone with Dave Mark

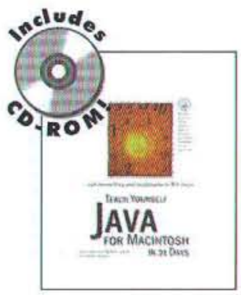
Easy-to-follow introduction for beginning programmers and Webmasters. Takes you through the core concepts of Java. Includes: Object-oriented techniques, unique features such as garbage collection, and basic programming concepts such as working with variables, threads and classes. Learn to apply this knowledge to write original Java applets for Web pages. Includes CD-ROM.

List Price $34.95 Our Price **$31.45** (BLJAVA)

Java Language API SuperBible
by Daniel Groner, Todd Sundtsed, Casey Hopson and Harish Prabandham

Covers Java 1.1. Provides hundreds of concrete source code examples. Sample projects introduced & assembled in each chapter.

List Price $59.99 Our Price **$53.99** (BJLAS)



Teach Yourself Java for Macintosh in 21 Days
by Laura Lemay and Charles L. Perkins with Timothy Webster

Add interactivity and multimedia to Web pages! A step-by-step guide to make your Website come alive. Learn the basics of programming Java applets and the concepts behind the Java language. Includes CD-ROM with a limited version of Roaster, the first commercial, integrated applet development environment for Java for the Macintosh!

List Price $40 Our Price **$36** (BJAVAMAC)

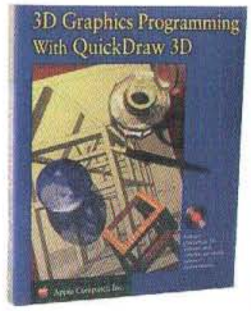
Intranet Web Development:
Enterprise Alternatives to Client/Server Computing
by Hayden Development Team

Learn why traditional client/server computing is being replaced by new WWW technologies. Discusses WWW application development with the Microsoft Visual Script enterprise application developer in mind. Learn techniques for the conversion of existing Visual Basic, C++ and PowerBuilder Applications to the new WWW platform.

List Price $49.99 Our Price **$45** (BINTWD)

WAIT... There's More!

PRODUCT	CODE	LIST PRICE	OUR PRICE
CGI By Example	SSGIBE	34.99	31.45
NetObject Fusion Handbook	BNETOFH	50.00	45.00
Learn HTML on the Mac	BLHTML	29.95	26.95
Mastering Netscape 2.0 for Mac, 2nd Ed.	SMASNET4	40.00	36.00
Infini-D Revealed	SINFDREV	45.00	40.50
DeBabelizer: The Authorized Edition	BDEBTAE	45.00	40.50
Sex, Lies and Video Games	BSEX	34.95	31.46

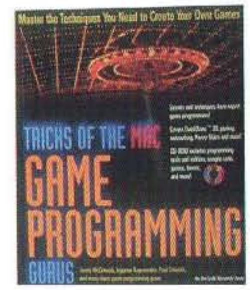


3D Graphics Programming Using QuickDraw 3D

by Apple Computer, Inc.

Incorporate spectacular 3D graphics into your applications. Explore QuickDraw 3D, a revolutionary graphics extension to the Mac OS for Power Macintoshes. CD contains the complete QuickDraw 3D system itself and a complete database of the QuickDraw 3D API, allowing you instant access to the hundreds of graphics calls via a fast viewing engine. Book/CD-ROM, 640 pages.

List Price $39.95 Our Price **$35.96** (B3DGRAP)

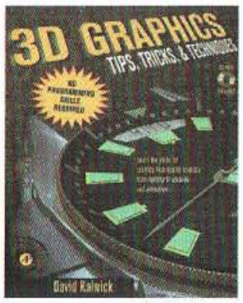


Tricks of The Mac Game Programming Gurus

by McCornack, Ragnemalm, Celestin, et al.

For beginning to expert game programmers. Complete overview of all the necessary components of game programming on the Macintosh. Packed with valuable tools, utilities, sample code, CodeWarrior™ Lite and game demos. QuickDraw 3D and Power Mac optimization and inside info on how Glypha III was created. Hundreds of tried-and-true tricks, tips, and insider secrets from well-known Mac game programming experts.

List Price $50 Our Price **$45** (BTRICKS)

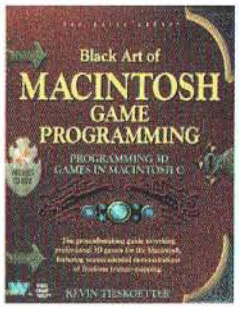


3D Graphics-Tips, Tricks, & Techniques

by David Kalwick

Written from a user's point-of-view, this book covers all the important 3D techniques including lighting, textures, animation, camera angles, and perspectives. Through easy-to-understand examples you'll learn how to create high-quality 3D graphics. Includes a Windows/Macintosh CD-ROM featuring tutorials, 3D examples from the book, and more.

List Price 34.95 Our Price **$31.46** (B3DGTTT)

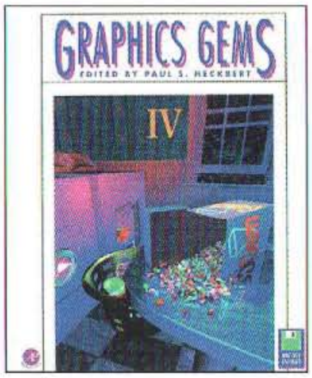


Black Art of Macintosh Game Programming

by Kevin Tieskoetter

Develop your own 3D games in C on the Mac. Includes CD with project files for both Symantec C and Code Warrior. Create freeform texture-mapped games and polygon graphics. Control dynamic source code - all compatible as native to the Power Mac. Write directly to the screen, bypassing QuickDraw.

List Price $39.99 Our price **$35.99** (BBLACK)



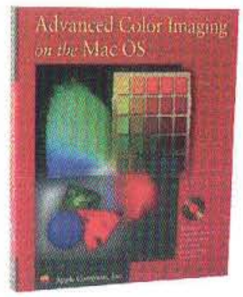
Graphic Gems V

Edited by Alan W. Paeth

Loaded with practical tools for implementing new ideas and techniques, to offer working solutions to real programming problems. Contains over 40 new gems in ellipses, splines, Bezier curves, and ray tracing – displaying the most recent and innovative techniques in graphics programming. Includes a disk with source code from all five volumes. Available in both IBM and Macintosh versions. CONTENTS: Algebra and Arithmetic Computational Geometry, Modeling and Transformation, Curves and Surfaces, Ray Tracing and Radiosity, Halftoning, Image Processing, and Utilities.

List Price $49.95 Our Price **$44.95** (BGEMS5)

Includes Disk!



Advanced Color Imaging on the Mac OS

by Apple Computer, Inc.

Enhance your software's color capabilities with step-by-step instructions. Augment the color support supplied with QuickDraw, and QuickDraw GX. Use the Pallette Manager to get the best colors on limited displays. Match colors between screens and input/output devices (scanners & printers). CD includes a complete reference information in both QuickView and Acrobat formats. Plus, a sample application demonstrating ColorSync programming techniques.

List Price $36.95 Our Price **$33.25** (BADVCI)

Order Toll-free
800-MACDEV-1
(800-622-3381)

WAIT... There's More!

Here are more products. For full product descriptions please see our Web site, or feel free to call, fax, or E-mail us. Our prices on books are at least 10% off list price.

Development Environments

PRODUCT	CODE	PRICE
AppleScript Applications: Building Applications w/FaceSpan	BAPSCAP	$31.45
Basic for the Newton - Programming using NS BASIC	BNEWT	32.35
C++ Programming with CodeWarrior	BCPPCW	32.35
C/C++ SDK User's Guide	BCPPUSER	29.00
CodeWarrior Inside PowerPlant	BINSPP	34.95
CodeWarrior Software Development using PowerPlant	BCWSWDEV	31.45
Dan Shafer Presents the Power of Prograph CPX	BDANPRO	19.95
FORTRAN SDK User's Guide	BFORTUSER	34.95
Inside CodeWarrior Book	BINSCW	34.95
Last Resort Programmers Edition	BLSTRSRT	74.95
Learn C on the Macintosh, 2nd Edition	BLEARNC2	31.45
Mastering the Think Class Library	BMASTERTCL	26.95
Metrowerks CodeWarrior Programming	BCWPROG	35.95
Objective-C - Object-Oriented Programming	BOBJCOOPT	34.15
Presenting Magic Cap	BPRESMAGIC	15.25
Real World Apple Guide	BREALWLD	35.95
Symantec C++ Programming	BSYMCPP	39.50
Taligent's Guide to Designing Programs	BTALIGENT	17.55
Visual Programming with Prograph	BVISPRO	30.60
Wireless For The Newton	BWIRELESS	31.45

Hardware

PRODUCT	CODE	PRICE
Apple CD-ROM Handbook	BCDHAND	14.36
Designing Cards & Drivers for the Macintosh	BCARD	26.96
LaserWriter Reference	BLASERREF	17.96
PCI System Architecture 3rd Edition	BPCISYS	31.46
PowerPC System Architecture	BPPCARCH	31.46

Internet Related

PRODUCT	CODE	PRICE
1994 Internet White Pages	B94WHITE	26.95
Active Java	BACTJAVA	23.36
America Online for Dummies	BAOLDUM	17.95
CGI By Example	BCGIBE	31.45
Computer Privacy Handbook	BPRIV	22.45
E-Mail Essentials	BEMAILE	22.45
Elements of E-Mail Style	BEMAIL	13.45
Hooked on Java	BHJAVA	26.95
Instant Internet Guide	BINSTANT	13.45
Internet Book	BTHENET	22.50
Internet for Dummies 2nd Edition	BNETDUM2	17.99
Internet for Dummies Quick Reference	BDUMQCK	8.05
Internet for Macs for Dummies	BNETDUM	17.95
Internet for Macs for Dummies Bestseller Edition	BIFMFDBE	35.99
Internet Power Tools	BPWRTOOL	36.00
Internet Publishing with Adobe Acrobat	BIPWAA	36.00
Internet Secrets	BSECRET	35.99
Internet, The, Deluxe Edition	BNETDELUX	31.50
Intranet Web Dev.: Enterprise Alternatives to Client/Server	BINTWD	44.99
Java Essentials for C/C++ Programmers	BJAVAESSEN	17.95
Java in a Nutshell	BJAVANUT	13.45
Java Language API SuperBible	BJLAS	53.99
JavaScript for Macintosh	BJAVASCRIPT	40.50
Learn HTML on the Macintosh	BLHTML	26.95
Learn Java on the Macintosh	BLJAVA	31.45
Mastering Netscape 4.0 for Macintosh, Second Edition	BMASNET4	36.00
More Internet for Dummies Starter Kit	BDUMNET	17.95
Mosaic for Dummies	BMOSDUM	17.99
Net Chat	BNETCHAT	17.00
NetObjects Fusion Handbook	BNETOFH	45.00
Netscape Navigator	BNETNAV	26.95
Netscape Navigator Starter Kit	BNETNSK	31.49
Perl Quick Reference	BPERLREF	17.99
Planning and Managing Web sites	BPLANWEB	35.96
Providing Internet Services	BPROVNET	31.46
Publish it on the Web	BWEBPUB	31.46
TCP/IP Vol 1-Vol 2 Bundle	BTCP12BNDL	99.00
Teach Yourself Java in 21 days	BJAVAMAC	36.00
Underground Guide to Telecommuting	BUNDER	22.45
Web Head Mac Guide	BWEBHEAD	22.45
Web Page Scripting Techniques	BWEBPST	45.00
Web Weaving	BWWEAV	22.45
Webmaster Macintosh	BWEBMAS	26.95

Scripting and Solutions

PRODUCT	CODE	PRICE
AppleScript Applications: Building Apps with FaceSpan	BAPSCAP	31.45
Applied Macintosh Scripting	BAPPLIED	31.45
Complete AppleScript Handbook	BAPLSCRHB	31.50
Complete HyperCard 2.2 Handbook	BHYPCRD2	31.50
Danny Goodman's Apple Guide Starter Kit	BDGAGSK	31.46
HyperCard Stack Design	BHYPSTA	19.95
HyperTalk 2.2: The Book	BHYPTAL	31.50
JavaScript for Macintosh	BJAVASCRPT	40.50
Perl Quick Reference	BPERLREF	17.99
Real World Apple Guide	BREALWLD	35.95

Technical Reference

PRODUCT	CODE	PRICE
Active Java	BACTJAVA	23.36
Apple CD-ROM Handbook	BCDHAND	14.36
Art of Human Interface Design	BAHID	29.65
Black Art of Mac Game Programming	BBLACK	35.99
C++ for Dummies	BCPPDUM	17.95
C for Dummies Vol. 1	BCDUM	17.95
Developing Object Oriented Software for the Macintosh	BDEVOB	26.06
Essential OpenDoc	BESOD	35.95
Extending the Mac Toolbox	BETMT	22.46
Foundations of Macintosh Programming	BFOUND	35.96
Fragment of Your Imagination	BFRAG	35.96
Guide to Macintosh Software Localization	BLOCALIZ	24.26
Guide to Macintosh System 7.5	BSYS7.5	22.50
How to Write Macintosh Software	BWRITE	26.05
Inside AppleTalk	BAPTALK	31.45
Inside the Macintosh Communications Toolbox	BCOMM	22.45
LaserWriter Reference	BLASERREF	17.96
Learn C++ on the Macintosh	BLRNCPP	35.05
Learn C on the Macintosh, 1st Edition	BLEARNC1	31.45
Learn C on the Macintosh, 2nd Edition	BLEARNC2	31.45
Mac Programming for Dummies	BMACDUM	17.95
Macintosh C Programmer Primer Volume 1	BCPRIM1	24.25
Macintosh C Programmer Primer Volume 2	BCPRIM2	24.25
Macintosh OLE2 Prog. Reference Working with Objects	BOLE2	40.45
Macintosh Pascal Programming Primer Volume I	BPASCPRI	24.25
Macintosh Programming Secrets	BPSECRET	28.76
Macintosh Programming Techniques	BPTECH	31.95
More Mac Programming Techniques	BMORETECH	34.50
Network Frontiers Bundle	BNETFB	59.95
NeXTStep Development Tools & Techniques	BNEXTDTT	27.85
NeXTStep Development Tools & Techniques	BNEXTDTT	27.85
NeXTStep General Reference	BNEXTGR	40.45
NeXTStep Programming Interface Summary	BNEXTPIS	27.85
NeXTStep User Interface Guidelines	BNEXTUIG	22.45
Newton Programming Guide	BNEWTPGUID	40.46
Object Oriented Programming Design	BOOPRODES	20.66
Optimizing PowerPC Code	BOPTPPC	35.96
PostScript Language Reference	BPSLANREF	29.66
Powerbook: Digital Nomad's Guide	BPBTDNG	22.46
PowerPC Programmer's Toolkit	BPPCPT	40.50
Programming Introduction to the Macintosh Family	BFAMILY	22.46
Programming for System 7	BSYS7	24.25
Programming Primer Macintosh Volume 1	BPRIMMAC	34.15
Programming QuickDraw	BPROQDRAW	24.25
Programming Starter Kit	BPROSTART	40.50
Programming with AppleTalk	BPROAT	22.45
QuickTime - Official Guide for Macintosh Users	BQTGUIDE	45.00
QuickTime Starter Kit for the Macintosh	BQTSKIT	40.50
Real World Apple Guide	BREALWLD	35.95
ResEdit All Night Diner	BRESDINE	22.45
ResEdit Complete, 1st Edition	BRESED1	31.45
ResEdit Complete, 2nd Edition	BRESED2	31.45
ResEdit Reference	BRESEDREF	26.96
Software by Design: Creating User Friendly Software	BDESIGN	26.95
Symantec C++ for the Macintosh: The Basics	BSCFTMTB	31.46
Teach Yourself Macintosh C++ in 21 Days	BCPP21D	26.99
Technical Introduction to the Macintosh Family	BTITTMF	24.26
Tog on Software Design	BTOG	26.95
Wireless for the Newton Development for Mobil Comm.	BWIRELESS	31.45
Writing Localizable Software	BLOCAL	24.25

Miscellaneous

PRODUCT	CODE	PRICE
Adobe Premiere for the Macintosh	BPREM	44.95
America Online for Dummies	BAOLDUM	17.95
Art of Human Interface Design	BAHID	29.65
CD ROM Guide to Multimedia Authoring	BCDMULTI	40.45
CompuServe for Dummies	BCSDUM	17.95
Creating Interactive CD-ROM	BINTERCDR	35.95
Cyberpunk Handbook	BCYBPUNK	8.95
Danny Goodman's Macintosh Handbook	BGOODHB	26.95
eWorld: The Essential Guide	BEWORLD	9.95
FrameWorks Source Code Disk	MTFWSC	9.95
FrameWorks Magazine Back Issue	MTFWBACK	8.00
Global Interface Design	BGLOBAL	32.35
Graphic Gems 2	BGEMS2	44.95
Graphic Gems 4	BGEMS4	44.95
Graphic Gems V	BGEMS5	44.95
Infini D Revealed	BINFDRFV	40.50
Late Night with MacHack	BLATE	26.95
Mac Bathroom Reader	BBATH	11.70
Mac Screamer: The Ultimate Macintosh Supercharging Kit	BSCREAM	31.50
Macintosh Crash Course	BCRASH	26.95
MacTech Back Issues	MTBACKISS	10.00
Macworld Ultimate Macintosh Book	BULTMAC	35.95
Managing AppleShare & Workgroup Servers	BMAWS	26.95
MADACON '93 CD-ROM	SMADA93	9.95
Multimedia Authoring: Building and Developing Documents	BMMAUTH	31.45
Multimedia Starter Kit for Macintosh	BMMSTART	27.00
Profit from Experience	BPROFIT	22.45
Sad Macs, Bombs and Disasters	BSADMAC	22.45
Stupid Mac Tricks	BSTUPIDMAC	17.95
The Software Developer's & Marketer's Legal Companion	BSDAMLC	33.26
Tricks of the Mac Game Gurus	BTRICKS	45.00
Zen and the Art of Resource Editing	BZAAORE	27.00

All entries in this index are alphabetized. For your convenience the product names are **bold**, book names are *italicized* and company names are in plain text.

Unlock the power of PowerPC microprocessors.

Motorola's optimizing C/C++ SDK, PowerPC™ Edition for Mac OS, is designed specifically for developers creating applications and tools for computer systems running Mac™ OS. The Motorola compiler works with Metrowerks CodeWarrior, Symantec Project Manager and Apple Macintosh Programmer's Workshop (MPW®) to offer a powerful set of development tools within a complete development environment.

The SDK features the Kuck & Associates Preprocessor (KAP™) — a powerful code restructuring tool designed to improve performance across a broad range of application programs. The Motorola C/C++ SDK delivers world-class code optimization for any specific PowerPC microprocessor, while ensuring compatibility across the entire PowerPC family of microprocessors.

To find out how Motorola can help you unlock the power of PowerPC microprocessors, visit our web site at http://www.mot.com/PowerPC/ or call us at 1-800-345-8384 or 1-512-891-2229.

PowerPC™

Mac® OS

MOTOROLA

What you never thought possible.®

Still the codewriter's best friend.

CodeWarrior. We're loyal, we're tough, and we're ready to take on Rhapsody. In '94, we helped you master writing code for the PowerPC™. Now you're entering new territory. Do it with development tools that are as familiar as your own backyard: the powerful, proven CodeWarrior 12 Integrated Development Environment, with hybrid C++/Objective C compilers. You want Apple's new Operating System for symmetric multiprocessing, preemptive multitasking and protected memory. You want CodeWarrior 12 for proven performance, unrivaled customer service and dedicated support. Take us home - we're loyal, tough and completely housebroken.

CodeWarrior.
Mark your territory.

metrowerks

MacMall
1.800.222.2808

MacConnectio
1.800.986.2257 V10